NAVAL POSTGRADUATE SCHOOL
Monterey, California

DISSERTATION

MODELING SINGLE-EVENT TRANSIENTS
IN
COMPLEX DIGITAL SYSTEMS

by

Kenneth A. Clark

June 2002

Dissertation Supervisors: Herschel H. Loomis, Jr.
Alan A. Ross

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2002	**3. REPORT TYPE AND DATES COVERED** Doctoral Dissertation	
4. TITLE AND SUBTITLE: Modeling Single-Event Transients in Complex Digital Systems			**5. FUNDING NUMBERS**
6. AUTHOR(S) Kenneth A. Clark			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT

A methodology to determine the effect of single-event transients (SETs) on complex digital systems has been developed. This methodology is based on the SET-state-transition model. This model breaks the complex digital system down into five states. These states are the error-free/transient-free state, the logic-gate-transient state, the single-event-upset (SEU) state, the output-driver transient state, and the failure state. The state-transitional probabilities of the model are determined by SET generation modeling, SET propagation modeling, and SEU propagation modeling. SET generation and propagation are primarily modeled using SPICE. SEU propagation modeling is accomplished using a combination of VHDL fault-injection modeling and mode-dependent (or instruction-based for a processor) register-usage analysis.

To verify this methodology, the SET tolerance of a 16-bit RISC microprocessor, the KDLX, was predicted. The transitional probabilities for this processor were determined, and the effective cross-section of the processor for three different test programs was predicted. Laser testing was performed on the KDLX to validate the predicted transitional probabilities. Heavy-ion testing was performed to validate system-level predictions. The results from the heavy-ion testing show that the methodology accurately predicts the saturated effective cross-section of a complex digital system.

14. SUBJECT TERMS single-event-transients; single-event upsets; single-event effects; transient fault propagation;			**15. NUMBER OF PAGES** 183
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

MODELING SINGLE-EVENT TRANSIENTS IN COMPLEX DIGITAL SYSTEMS

Kenneth A. Clark
B.S., The University of Virginia, 1990
M.S., The Johns Hopkins University, 1993

Submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2002

Author: _____
Kenneth A. Clark

Approved by:

_____
Herschel H. Loomis, Jr., Professor
Dept. of Electrical & Computer Engr.
Dissertation and Committee Supervisor

_____
Alan A. Ross, Professor
Space Systems Academic Group
Dissertation Supervisor

_____
Douglas J. Fouts,
Associate Professor,
Dept. of Electrical & Computer Engr.

_____
Todd Weatherford,
Assistant Professor,
Dept. of Electrical & Computer Engr.

_____
George E. Price,
Engineer,
Naval Research Laboratory (retired)

Approved by: _____
Jeffrey B. Knorr, Chair, Department of Electrical & Computer Engr.

Approved by: _____
Carson K. Eoyang, Associate Provost for Academic Affairs

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

A methodology to determine the effect of single-event transients (SETs) on complex digital systems has been developed. This methodology is based on the SET-state-transition model. This model breaks the complex digital system down into five states. These states are the error-free/transient-free state, the logic-gate-transient state, the single-event-upset (SEU) state, the output-driver transient state, and the failure state. The state-transitional probabilities of the model are determined by SET generation modeling, SET propagation modeling, and SEU propagation modeling. SET generation and propagation are primarily modeled using SPICE. SEU propagation modeling is accomplished using a combination of VHDL fault-injection modeling and mode-dependent (or instruction-based for a processor) register-usage analysis.

To verify this methodology, the SET tolerance of a 16-bit RISC microprocessor, the KDLX, was predicted. The transitional probabilities for this processor were determined, and the effective cross-section of the processor for three different test programs was predicted. Laser testing was performed on the KDLX to validate the predicted transitional probabilities. Heavy-ion testing was performed to validate system-level predictions. The results from the heavy-ion testing show that the methodology accurately predicts the saturated effective cross-section of a complex digital system.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGEMENTS

This dissertation has proven to be one of the most challenging endeavors I have attempted. It is also among my proudest accomplishments. There are many people who have helped me along the way. They have provided technical guidance, encouragement and support.

I would like to thank the members of my dissertation committee. First and foremost, this would not have been possible without the patience of my committee and their willingness to participate in numerous teleconferences along the way. Herschel Loomis, as the Dissertation Chairman and Co-Advisor, has provided me with excellent feedback and comments during the process. Alan Ross, also a Co-Advisor, has helped me to "stay the course," making sure that the research did not get off-track from what we set out to accomplish. He also provided valuable critique during the drafting of the manuscript, which resulted in a much better final product. The same could be said of Douglas Fouts, who also gave me advice during the integrated circuit design process of the KDLX processor. Todd Weatherford provided excellent technical guidance in the area of semiconductor physics and, as a result, this dissertation is much more complete in that area. George Price, who was my supervisor at the Naval Research Laboratory when I returned from the Naval Postgraduate School, provided the initial suggestion for the subject of this dissertation and supported my research at the Lab.

I would like to thank my colleagues presently or formerly at the Naval Research Laboratory that helped me in my research. Steve Buchner provided encouragement, valuable technical feedback and was indispensable during the laser tests. Dale McMorrow also assisted in the laser tests. Paul Marshall and Art Campbell assisted me during the heavy-ion testing. Tim Meehan provided valuable technical advice and assistance as I was designing the KDLX processor and test system.

I am also grateful to my family. This dissertation would not have been possible were it not for the willingness of my wife, Lynn, to move our young family to Monterey, CA to begin my studies. Mary Lynn, Kevin and Justin, my children, have helped to keep my work on this dissertation in proper perspective with the rest of my life.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. ON-BOARD PROCESSING REQUIREMENTS OF SATELLITES

The planned capabilities of many satellites under development today are creating tremendous requirements for on-board processing. The Astrolink and DIAMANT satellites are good examples of this. The Astrolink satellites will provide an "ATM[1] router in space" [1]. Each satellite will be capable of providing 100 Mbits/sec access for each user and an aggregate bandwidth of 6 Gbits/sec. For each received ground transmission, the on-board processing circuitry must extract the ATM cells, create a virtual connection, and route the cells to the appropriate modulator and antenna beam aimed at the desired destination. This is a very ambitious level of complexity for satellite electronics. The total gate count for the electronics is approximately 750 million [2].

The DIAMANT satellites will provide high-resolution multi-spectral images of the earth. At the heart of the satellites is the Multi-Spectral high-Resolution-System (MSRS) sensor. This advanced sensor will provide imagery from 12 narrow-spectral bands in the very-near-infrared (VNIR) spectral range. The spatial resolution of the imagery is approximately 5 meters [3]. At this resolution, a scene of size 50x700 km requires 84 Gbits of on-board data storage per band. Additionally, the downlink data rate is limited to about 280 Mbits/sec. As a result, on-board processing is necessary to compress the imagery data [4]. Greater on-board processing directly improves the total number of images the satellite will be able to handle. Thus, for the DIAMANT satellites, it is desirous to have the maximum amount of on-board processing as allowed by the size, mass, and power capabilities of the satellite.

B. TECHNICAL PROBLEM DESCRIPTION

These on-board processing requirements create a challenge to the satellite electronics designers: they must balance the processing requirements with the requirement to operate reliably in the space radiation environment. To do this, the designers not only must provide the necessary processing capability, but they also must

[1] ATM stands for Asynchronous Transfer Mode.

assure that the potential effects of radiation will not prevent proper operation of the electronics.

The risks associated with the effects of radiation in space are well understood. These risks come from total-dose degradation, dose-rate effects, and single-event-transient effects. An accepted characterization method for total-dose testing of electronics devices exists [5]. The same can be said about dose-rate-effects testing [6, 7]. Most areas of single-event-transient testing are also well understood. For example, testing for single-event latchup and single-event upsets in simple devices (e.g., memories) has a standard approach agreed upon in the radiation-effects community [8]. The missing piece to the puzzle, though, is the characterization of single-event transients (SETs) in complex digital systems.

C. OBJECTIVE OF RESEARCH

The objective of this research is to formulate, verify, and validate a methodology to characterize the single-event-transient tolerance of complex digital systems. A complex digital system is defined as a system that contains more than one functional mode and is comprised of both combinational logic and memory elements. By this definition, a complex digital system can range from a state machine to the latest processor. The system may be a single chip, or it may consist of many chips. This methodology must be applicable to this range of systems. It must be suitable to all implementations of digital systems, which include field-programmable gate arrays, standard-cell application-specific integrated circuits, and off-the-shelf processors. In all cases, the methodology must account for the two key aspects of a complex digital system: that it contains multiple functional modes, and that it contains both combinational logic and memory elements.

D. TECHNICAL APPROACH

The formulation of this methodology is based on an SET-state-transition model that accounts for the unique aspects of a complex digital system. The model defines the transitional probabilities necessary to go from a fault-free state to a failure state. These transitional probabilities are predicted by a combination of modeling and simulation. The verification of the methodology was accomplished by determining the transitional

probabilities for a candidate complex digital system. Once the transitional probabilities had been determined, the SET-state-transition model was used to determine the probability of the system going from the fault-free state to the failure state.

A 16-bit, 5-stage-pipeline RISC[2] microprocessor was the candidate complex digital system. It was fabricated through the MOSIS integrated-circuit fabrication service using standard-cell design techniques. This approach provided a hardware-description-language (HDL) definition of the microarchitecture of the processor, a SPICE transistor-level description of the individual elements, and the parametric test results of the MOSIS foundry run. This allowed for thorough simulations to determine the transitional probabilities.

Validation was accomplished by performing radiation testing to compare the predicted upset rates with the measured upset rates. Figure 1 summarizes the steps taken to validate the methodology. First the methodology was formulated. It was verified by predicting the system upset rate of the RISC processor. Radiation testing was then performed on the device. The methodology was validated by agreement between the measured upset rate and the predicted upset rate.

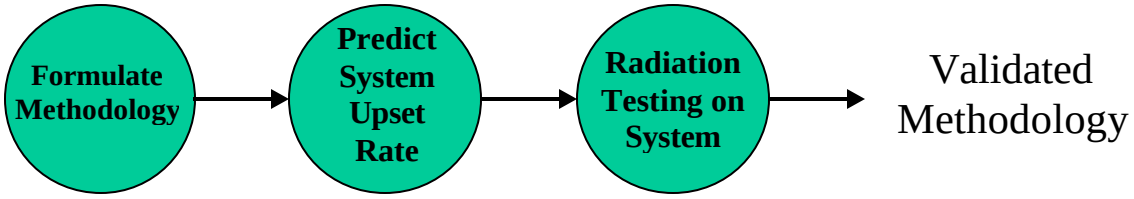


Figure 1. Methodology Validation Path

E. DISSERTATION ORGANIZATION

This dissertation is organized in a similar manner to the steps shown in Figure 1. **Chapter II – Methodology Description** introduces the SET-state-transition model to develop the methodology. **Chapter III – Modeling Approach** describes in more detail how the various transitional probabilities are modeled and determined. This includes SET generation, SET analog propagation, SET logic propagation, SET clock-edge effects, and SEU propagation. In **Chapter IV – Modeling and Simulation**, the

methodology and modeling approach described in previous chapters are performed on the RISC microprocessor. A system-level prediction of the microprocessor is provided. **Chapter V – Modeling Validation** documents the results of the radiation testing. This testing includes both laser testing and heavy-ion testing. A comparison between the measured upset rate and the predicted upset rate is discussed. **Chapter VI –Conclusion** summarizes the formulation, verification, and validation of the methodology. It describes how the methodology can be simplified to provide the "90% solution." It also shows how this methodology can be extended to other implementations of complex digital systems. Original contributions to the state-of-the-art are discussed, and areas for further investigation are suggested for future research.

[2] RISC stands for Reduced Instruction Set Computer.

II. MODELING METHODOLOGY

A. OBJECTIVE

The objective of this chapter is to describe a methodology to characterize the SET tolerance of complex digital systems. Methodologies currently used by both the radiation-effects community and the fault-tolerant-computing community are reviewed. An SET-state-transition model is then defined, and the methodology is developed from this model.

B. PREVIOUS WORK

Methodologies for determining the SET tolerance of complex digital systems generally approach the problem from two different perspectives: either injecting transients at the device level (through irradiation) and measuring the system impact, or injecting transients at the circuit level (through simulation) and tracing error propagation to the system level. The radiation-effects community has largely been responsible for developing the injection-by-irradiation methodology, and the fault-tolerant-computing community has been largely responsible for the injection-by-simulation methodology.

There have been many papers from the radiation-effects community about the SET tolerance of complex digital systems. Deb Newberry has written several papers on the results of testing a spaceborne 1750A processor system [9, 10, 11]. In these papers, the system consisted of processors, memory, and peripheral logic. One of twenty software programs was run. The results of the tests showed that it is possible for an error to propagate from one device to another in the system. It was also shown that the error rate for a processor system is a strong function of the test software used.

In Label [12], a different approach was used: the actual flight software was run on the system during the radiation test. In this case, the test methodology focused more on the validation of the planned flight configuration than on the full characterization the SET tolerance of the system.

In Kimbrough [13], the single-event-upset (SEU) performance of several R3000-based RISC processors was characterized. This paper acknowledged the difficulty of characterizing processors: “Determining the cross-section of a processor is complicated

by device architecture and test software. Physically, the microprocessor is made of different functional blocks with varying architecture. The cross-section is dependent upon how extensively the software checks the functional blocks." In spite of this acknowledgement, no attempt was made to provide these various cross-sections that are a function of the test software.

The methodology used in Koga [14] is the most thorough. A test plan is provided to determine the individual sensitivities of the functional elements of a processor system: "If we can test the SEU vulnerability of each functional element, the combined rate of SEU in space can be estimated from the program execution pattern. This 'macroscopic' (functional element as opposed to individual circuit) testing of many functional elements can be accomplished externally using the standard instruction sets (i.e., there is no need to obtain test circuits especially fabricated for microscopic SEU testing)." The three stages of testing a microprocessor are:

1. "... select an appropriate test method, using selection criteria, such as microprocessor architecture, operating speed, instruction formats, circuit design, and application software."

2. "... deduce the SEU cross-section as a function of LET for various memory elements and any other elements (using appropriate ground-test procedures and microprocessor element utilization factors during software executions)."

3. " ... using an appropriate physical model, we can combine data from step 2 with a radiation environmental model to compute upset rate in the environment."

For step 3, it is suggested that "... at the system level, power weights must be assigned to the individual element cross-sections when arriving at an overall system cross-section."

In summary, the methodologies from the radiation-effects community focus on injecting transients at the system level with radiation. In each case, it is recognized that the upset rate is a function of the software that is run during the test, but Koga [14] is the only one that provides a method to determine the cross-sections of the various functional elements within processors.

Papers from the fault-tolerant-computing community tend to focus on fault injection through simulation. The typical methodology is to inject a fault in the circuit design and determine if it propagates to the output. In Ghosh [15], a fault-injection methodology using a VHDL model is described. The approach allows for fault injection at various levels in a VHDL design: from the behavioral models down to the logic-gate-level VHDL descriptions. This methodology "involves the interception of signals and the corruption of the information present on the signal according to fault-injection times and error types."

In Cha [16], transient faults are injected at the analog level, where they propagate to the logic level. This paper defined a methodology that bridges the gap from an analog transient to the logic level. However, there was no attempt to tie the analog transient to a probability of occurrence.

In general, methodologies from the radiation-effects community seek to characterize the single-event-effect tolerance of a device given a fluence of incident ions. This characterization is usually made without much insight into the design of the device. The methodologies used by the fault-tolerant-computing community generally seek to evaluate how well a design operates given a transient fault has occurred. This transient fault can be at the analog level or the logic level, but the likelihood of a transient fault occurring is not considered. These methodologies from the two communities can complement each other. By combining the determination of the likelihood that an SET will occur from the radiation-effects community with the precise fault-propagation modeling from the fault-tolerant-computing community, a more complete methodology can be created.

C. SET-STATE-TRANSITION MODEL

1. Objective

The objective of the SET-state-transition model is to define the framework necessary to develop the methodology. It is a state-transition diagram that shows how an SET can cause the device or system to go from a fault-free state to a failure state. It is applicable to synchronous, asynchronous, and mixed-signal systems. In this dissertation, it is applied to a synchronous digital system.

2. Definitions

Prior to describing the SET-state-transition model, it is necessary to define some key terms. An **SET,** or **single-event transient,** is an unintended analog pulse. This dissertation focuses on SETs that are the result of incident heavy ions; however, the SET-state-transition model can apply to SETs resulting from other sources such as electromagnetic interference or power supply noise. A **single-event upset**, or **SEU**, occurs when an SET causes a bit-flip error in a memory element. **Failure** occurs when the component of interest causes an error in the external system.

3. Description

The SET-state-transition model is shown in Figure 2. It shows the states and transitional probabilities necessary to go from the fault-free state of the system to the failure state. The propagation states are described below:

S1: No SETs or SEUs: This is the normal, fault-free state of the system. The system will operate perfectly for as long as it remains here. From this state, an SET can cause a transition to states S2, S3, or S4. An SET on a memory element occurs with a transitional probability of $\beta 1$, causing the system state to be S3. An SET on a logic gate occurs with a transitional probability of $\beta 2$, causing the state to go to S2. Finally, an SET on an output driver occurs with a transitional probability of $\beta 3$ and causes the system state to be S4.

S2: Logic Gate Transient(s): In this state, one or more transients are propagating in the combinational logic. The transient or transients are the result of a single initial transient. If the fan-out of the logic gates in the path of propagation is greater than one, multiple transients may result. These transients can do three things: they can die out (transitional probability $\alpha 2$), be latched into a memory element (with transitional probability $\delta 1$), or propagate to an output driver (with transitional probability $\delta 2$). This assumes that the length of the pulse is less than one clock cycle.

S3: SEU: In this state, one or more SEUs are present in the system. This means that at least one of the memory elements in the system is in error. This can happen two ways: a transient can occur on the transistors that make up the memory element (causing

it to go directly from S1 to S3 with transitional probability β1), or the transient can be latched after propagating in the logic (causing a transition from S2 to S3 with transitional probability β2). Two state transitions are possible from S3: the SEU can be overwritten, bringing the state back to S1 (transitional probability α1), or the SEU can propagate to the output and cause an error to the external system, bringing the system state to S5 (transitional probability ε1). It should be emphasized that an SEU can propagate to the output without causing an error to the external system. A good example of this is an SEU in the address register of a processor that occurs when the processor is neither reading nor writing memory.

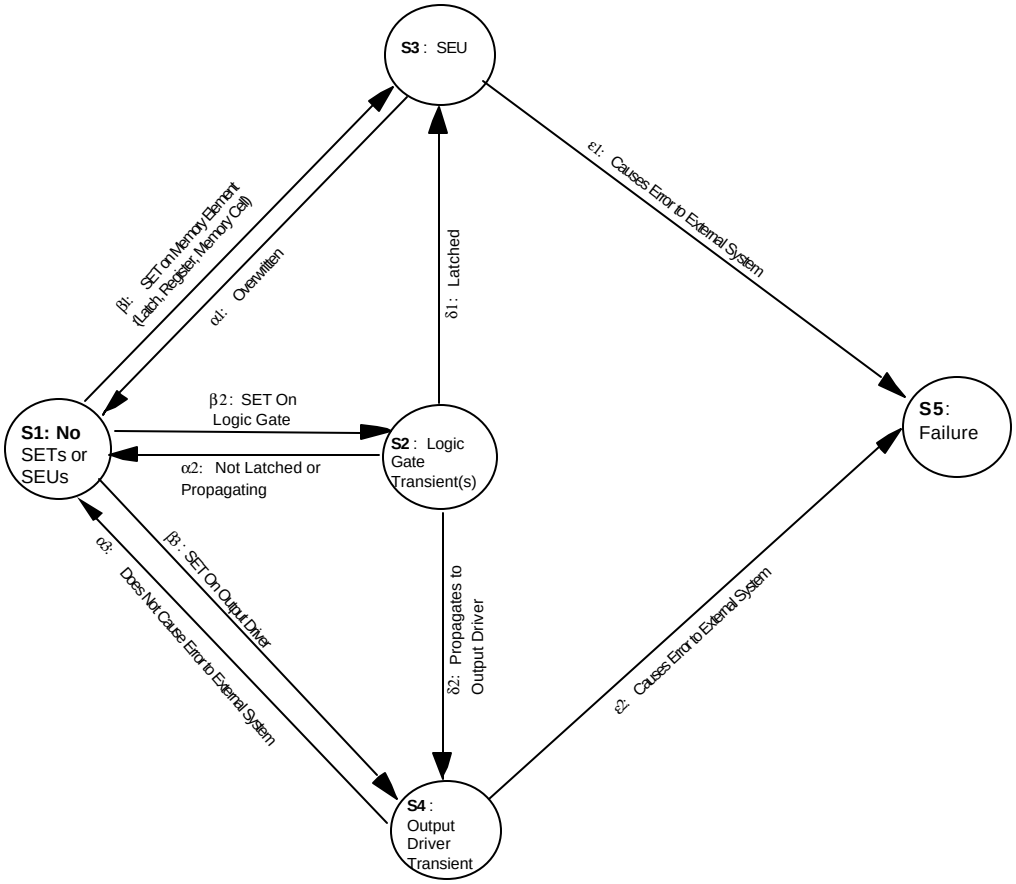


Figure 2. SET-state-transition Model

S4: Output Driver Transient: In this state, there is a transient on an output driver. If the transient does not cause an error to the external system, the state returns to S1 (transitional probability α3). If the transient does create an error in the external system (transitional probability ε2), the state goes to S5. It should be emphasized that whether or

not failure occurs depends on how the external system uses this output. For example, an output driver transient on an asynchronous-control signal of a processor, such as Write*, can immediately cause an error to the external system. In contrast, a transient on a data bus output driver that occurs when the processor is not reading or writing will not cause an error to the external system.

S5: Failure: In this state, the SET or resulting SEU has propagated to the output and caused an error in the external system. This marks the end of the simulation.

D. METHODOLOGY DEVELOPMENT

The transition model described in the previous section provides the basis for developing the methodology to characterize system level effects of SETs. Determining the overall system upset rate requires three steps:

Step 1: Determine Transitional Probabilities.

a. βn – SET generation probabilities – these probabilities are determined by modeling SET generation.

b. δn – SET propagation probabilities – these probabilities are determined by the three components SET propagation: SET analog propagation, SET logic propagation, and SET clock-edge effects.

c. εn – propagation to output – these probabilities are modeled with SEU Propagation Modeling (for $\varepsilon 1$) and SET analog propagation (for $\varepsilon 2$).

Step 2: Determine transitional probabilities for the given application.

Once the transitional probabilities have been determined for each functional state, the overall transitional probabilities for the given application can be determined with the equation below (for example, $\varepsilon 1$):

$$\varepsilon 1 = \Sigma\ \varepsilon 1(\text{in mode } n) \text{ x (mode } n \text{ duty cycle), for all } n. \qquad (2.1)$$

Step 3: Combine these transitional probabilities to account for the four possible paths from S1 to S5.

These are:

1. S1 -> S3 -> S5 [Probability = $(\beta 1)(\varepsilon 1)$],
2. S1 -> S2 -> S3 -> S5 [Probability = $(\beta 2)(\delta 1)(\varepsilon 1)$],
3. S1 -> S2 -> S4 -> S5 [Probability = $(\beta 2)(\delta 2)(\varepsilon 2)$],
4. S1 -> S4 -> S5 [Probability = $(\beta 3)(\varepsilon 2)$].

The overall probability of going from S1 to S5, which is the probability of failure, is the union of the above probabilities:

$$P(\text{failure}) = (\beta 1)(\varepsilon 1) + (\beta 2)(\delta 1)(\varepsilon 1) + (\beta 2)(\delta 2)(\varepsilon 2) + (\beta 3)(\varepsilon 2). \qquad (2.2)$$

E. METHODOLOGY APPLICATION

This methodology can apply to a wide range of complex systems, but first, several important aspects of the system must be considered. The output boundary of the system must be defined. There must be an exact definition of failure. The functional modes of the system must be well understood, because each functional mode must be considered in the determination of the transitional probabilities.

For a complex system that consists of complex subsystems, this SET methodology must first be applied to the subsystems. The results of the subsystem analyses are then used to determine the transitional probabilities at the system level.

THIS PAGE INTENTIONALLY LEFT BLANK

III. MODELING APPROACH

A. OBJECTIVE

The objective of this chapter is to develop the modeling approach to determine the transitional probabilities described in Chapter II. The modeling effort can be divided into five different areas: SET generation, SET analog propagation, SET logic propagation, SET clock-edge effects, and SEU propagation. Table 1 shows the relationship between the transitional probabilities and these modeling areas.

Transitional Probabilities	Modeling Areas
$\beta 1, \beta 2, \beta 3$	SET Generation
$\delta 1, \alpha 1$	SET Analog Propagation, SET logic Propagation, SET Clock-Edge Effects
$\delta 2$	SET Analog Propagation, SET Logic Propagation
$\varepsilon 1$	SEU Propagation
$\varepsilon 2, \alpha 2, \alpha 3$	SET Analog Propagation, SET Clock-Edge Effects

Table 1. Relationship Between Transitional Probabilities and Modeling Areas

B. SET GENERATION MODELING

1. Objective

The objective of SET Generation Modeling is to determine the transitional probabilities $\beta 1$, $\beta 2$, and $\beta 3$. For $\beta 1$ (SET on memory element), it is necessary to determine the probability of an incident ion depositing enough energy to cause the contents of the memory element to change. For $\beta 2$ and $\beta 3$, it is necessary to calculate the probability that an incident ion will result in an SET pulse with amplitude equal to *a* and pulsewidth equal to *pw*. This is denoted as $P_g(a, pw)$.

2. Underlying Theory

The probability of SET Generation is a function of how the electrical characteristics of a device are affected by the environment it is operating in. A CMOS inverter is shown in Figure 3. The input to the inverter is Gnd, and the output is driven to

Vdd. In this logical state, the NFET is in the "off" state and the PFET is in the "on" state. The drain voltage of the NFET is driven to Vdd volts by the PFET. This creates the depletion region shown at the drain of the NFET. The depletion region extends horizontally approximately one depletion layer width, *W*, to each side of the drain. For uniform doping, *W* is given by:

$$W = [(2e(V_0\text{-}V)/q)(N_a + N_d)/N_aN_d]^{1/2}\ [17], \qquad (3.1)$$

where e is the permittivity of silicon, V_0 is the contact potential, V is the applied potential, N_a is the acceptor concentration, N_d is the donor concentration. This creates a region that is sensitive to charge collection.

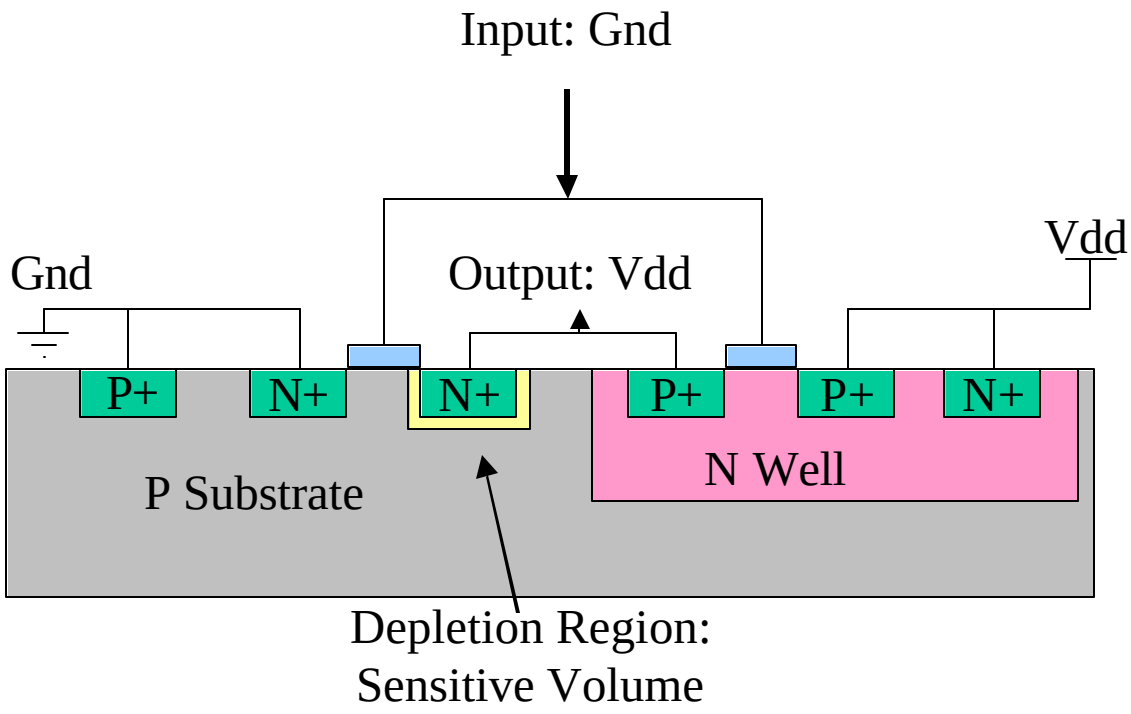


Figure 3. CMOS Inverter Cross-Section

The probability that an ion will strike this sensitive volume is a function of both the effective cross-section of this volume and the environment. The environment is often specified in terms of particle fluence versus Linear Energy Transfer (LET). The LET of an ion is the amount of energy that is transferred to the device per unit length. It is specified in units of MeV*cm^2/mg. For particles with a particular LET, the probability of an SET occurring within the sensitive region is given by:

$$\text{P(SET occurring with given LET)} = \sigma\Phi(LET), \qquad (3.2)$$

where σ is the cross-section of the sensitive region, specified in units of cm^2, and $\Phi(LET)$ is the fluence of particles with the given LET, specified in units of particles/cm^2.

If a particle strikes the sensitive region, a funnel of electron-hole pairs is created, as shown in Figure 4. The funnel length, L_f, is the linear distance of charge collection in the ion track. It is given by the following two equations:

$$\text{NFET: } L_f = (1 + (\mu_n/\mu_p)^k)^n, \qquad (3.3)$$

$$\text{PFET: } L_f = (1 + (\mu_p/\mu_n)^k)^n, \qquad (3.4)$$

where μ_n is the electron mobility, μ_p is the hole mobility, and the exponents k and n are determined empirically [18]. The number of electron-hole pairs created per unit length in silicon is given by the equation:

$$N = LET \text{ (MeV*cm}^2\text{/mg) x (density of Si(mg/cm}^3\text{))/3.6 eV [19]. (3.5)}$$

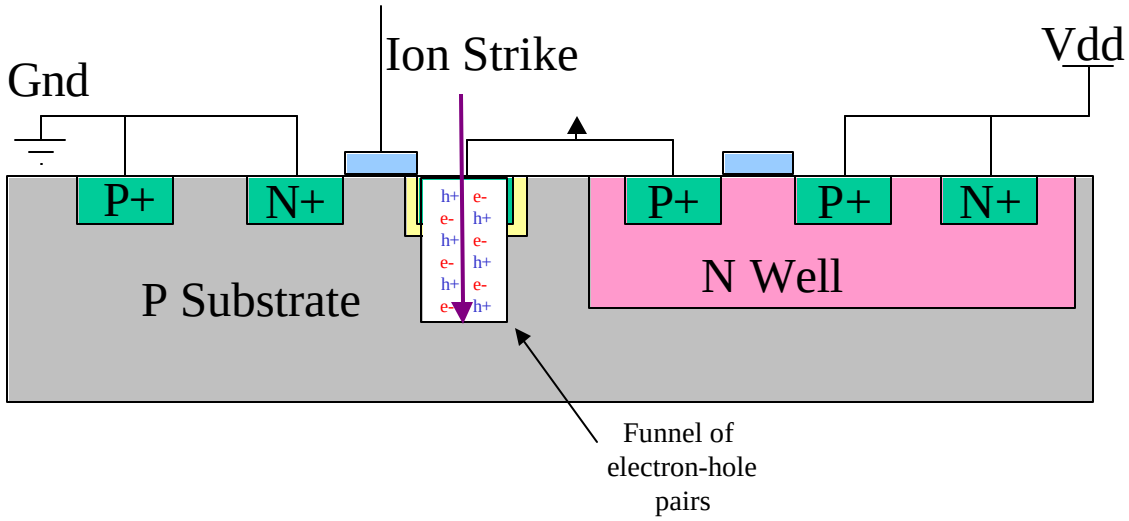


Figure 4. Charge Generation

Charge collection occurs as the free electrons are drawn to the drain (which is at Vdd), and the holes are drawn to the body (which is at Gnd) through the substrate. This is shown in Figure 5. At any particular plane within the funnel, the sum of the electron and hole drift currents is the net current flowing from the NFET drain to the substrate. This current reduces the NFET drain voltage (and output node voltage) below Vdd volts.

From Messenger[19], this charge deposition can be modeled as a double-exponential current pulse:

$$I(t) = I_0 \left[e^{-at} - e^{-bt} \right], \qquad (3.7)$$

where $1/\alpha$ is the collection time constant for the junction, and $1/\beta$ is the time constant for initially establishing the ion track, and I_0 is given by

$$I_0 = qmNE, \qquad (3.8)$$

where q is the charge of electron or hole, m is the ambipolar mobility of carriers, N is the number of electron-hole pairs generated per unit length (from equation 3.5), and E is the electric field component in the direction of the funnel. This assumes that diffusion and recombination are negligible during this time frame. Combining these equations and dividing by the funnel cross-section A and rearranging terms, gives:

$$I(t)/A = qm(N/A) \left[e^{-at} - e^{-bt} \right] E, \qquad (3.9)$$

Since $I(t)/A$ gives current density, $J(t)$, and $N/A = n = p$ (i.e., the carrier concentrations in units of electrons/cm^3 or holes/cm^3), then equation 3.9 can be rewritten as

$$J(t) = q(nm + pm) \left[e^{-at} - e^{-bt} \right] E. \qquad (3.10)$$

This has form similar to that of the drift current density equation from Streetman[17]:

$$J_x = q(nm_n + pm_p)e_x. \qquad (3.11)$$

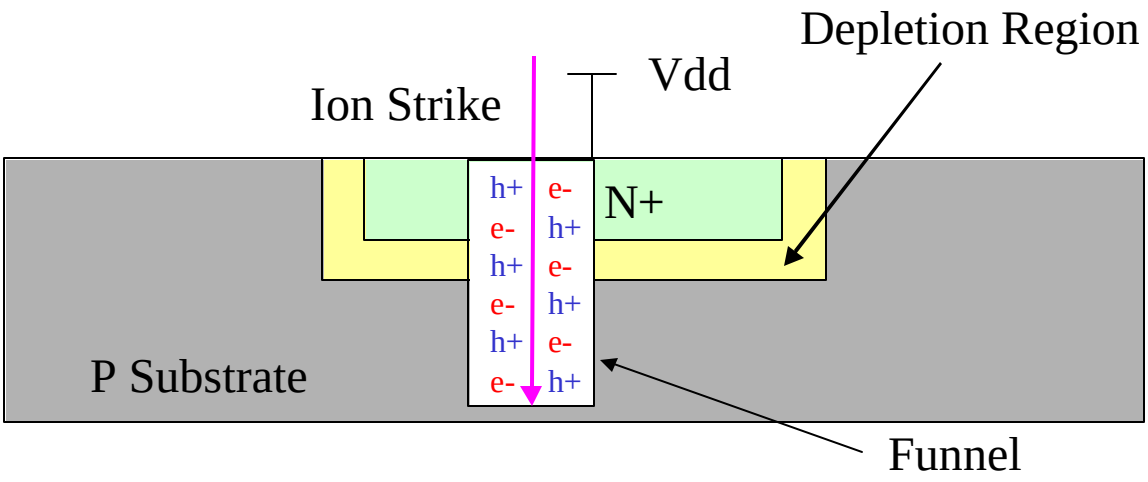


Figure 5. Charge Collection

This shows that the key funneling equation is simply the drift current density equation with two main differences. The first difference is the [e^{-at} - e^{-bt}] term, which describes the carrier concentrations decreasing as a function of time. The second difference is that equation 3-10 uses the ambipolar mobility for the electrons and holes. This assumes the carrier concentrations are ambipolar, which means the electron and hole concentrations within the funnel are changing at the same rate.

3. Previously Used Electrical Models

In previous papers [16, 20, & 21], the current pulse from equation 3.7 is modeled in SPICE with an independent current source with the output tied to the output node of a logic gate, as shown in Figure 6. The primary drawback with this method is that it represents the charge collection in a constant biased p+n junction. The problem, in this case, is that the bias of the p+n junction in question (the drain of the NFET) is not constant. It varies because the injection node voltage is changing as a result of the charge collection. A current source that is a function of the injection node voltage would be an improvement.

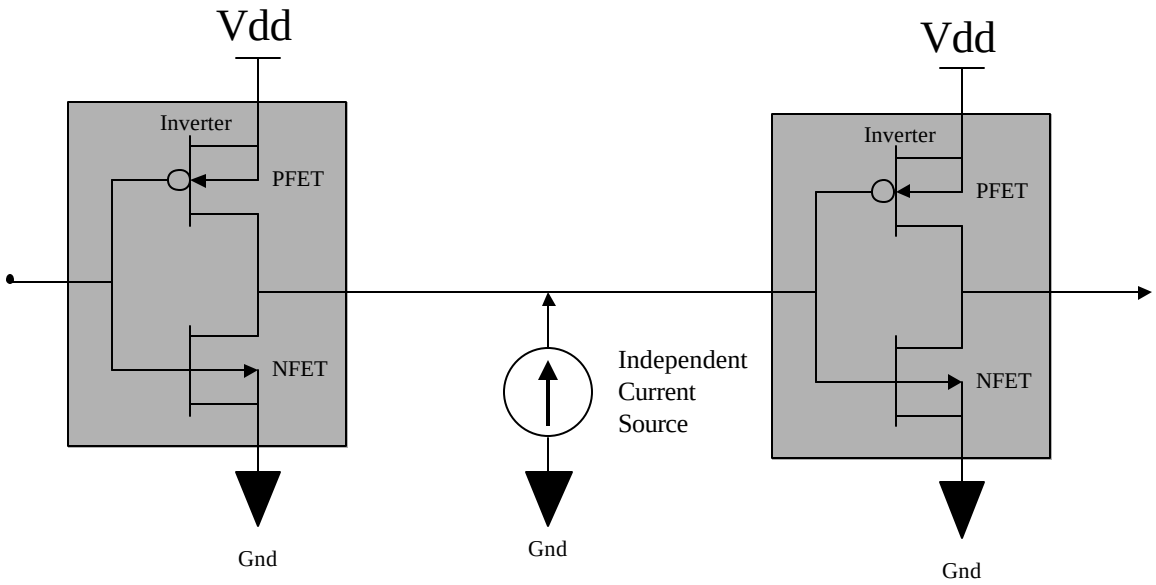


Figure 6. Current Source SET Injection

The injection source from Buchner[22] uses an NFET connected to Vdd with a resistor, as shown in Figure 7. The gate of the NFET is pulsed to inject charge onto the node. In practice, it is difficult to make the resulting current waveform look like the

desired double-exponential pulse. This is because the NFET has three modes of operation: cut-off, linear, and saturated; the transconductance of the NFET is different for each mode, making control of the current waveform difficult.

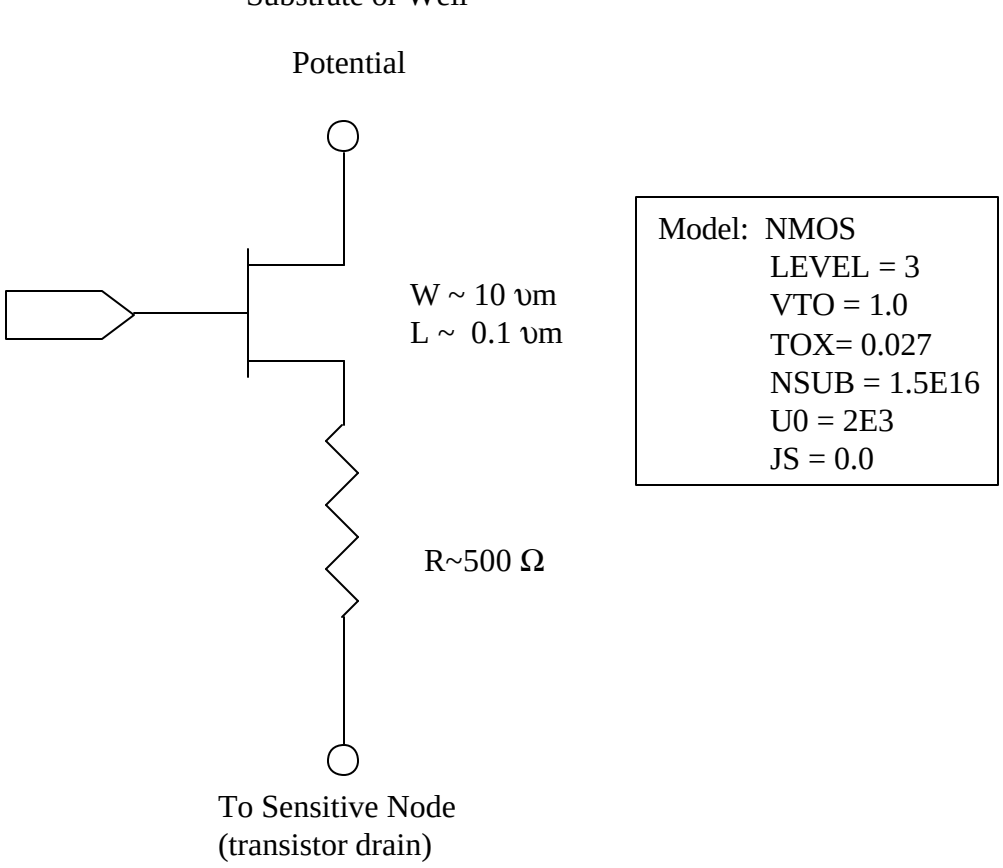


Figure 7. NFET SET Injection

4. SET Generation Modeling Approach

a. Determining the $P_{SET}(s,F)$

As described above, the probability of a particle with a given LET striking the sensitive region of a device is a function of both the device characteristics and the environment. Since a device often is used in multiple environments, the LET-dependent cross-section of the device alone is frequently used to define the SET susceptibility of the device. While it is necessary to multiply the particle fluence by the cross-section to determine the absolute $P_{SET}(\sigma,\Phi)$, determining the cross-section alone allows for relative

assessments of various devices independent of the environment. For this reason, the SET generation transition probabilities will be defined as LET-dependent cross-sections.

The cross-section of this sensitive region is the effective cross-sectional area of the depletion region of the drain of the sensitive MOSFET. The length (l_d) and width (w_d) dimensions of the drain are calculated from the drain area (AD) and drain perimeter (PD) parameters extracted from the layout of the device using the following equations:

$$AD = l_d\, w_d, \qquad (3.12)$$

$$PD = 2l_d + 2w_d. \qquad (3.13)$$

These dimensions are then used to determine the sensitive cross-section of the device using the following equations[23]:

$$\text{cross-section length: } l = l_d + 2W, \qquad (3.14)$$

$$\text{cross-section width: } w = w_d + 2W, \qquad (3.15)$$

$$\text{cross-section: } s = l\, w, \qquad (3.16)$$

where W is the depletion width from equation 3.1. These equations assume that diffusion does not add to the sensitive cross-section.

b. Electrical Modeling Approach

To overcome the shortcomings in the previous electrical models, the injection model must inject charge such that the amount of charge collected (injected) is not independent of the voltage on the node. Additionally, it is desired to have sufficient control of the current injection waveform. The model used is similar to that described in [16, 20, and 21], except I_o is not treated as a constant. Instead, it is modeled as a function of the node injection voltage using equations 3.7 and 3.8. This requires expressing the electric field, E, as a function of the node injection voltage.

Two cases must be considered: low-level injection and high-level injection. Low-level injection occurs when the excess carrier concentration within the funnel is lower than the majority carrier concentration yet higher than the equilibrium minority-carrier concentration. In contrast, high-level injection occurs when the excess

carrier concentration within the funnel exceeds the extrinsic doping levels and minority carrier concentrations [24]. The crossover point between low-level injection to high-level injection occurs when the excess carrier density within the funnel is equal the sum of the extrinsic doping level and the minority carrier concentration.

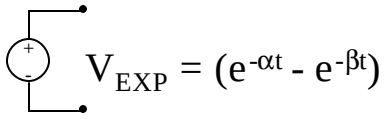
In low-level injection, the electric field is still defined by the junction. The electric field for equation 3.8 is given by the equation for the maximum value of the electric field within the junction [19, 25]:

$$E_0 = [(2q/e)^* (V_{node} - V_0)^*(N_a N_d)/(N_a+N_d)]^{1/2}, \qquad (3.17)$$

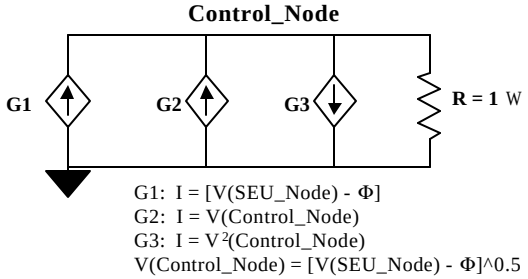
where e is the permittivity of the material and V_{node} = voltage of injection node. Substituting equation 3.17 into equations 3.7 and 3.8 gives:

$$I(t) = qmN\ [(2q/e)^* (V_{node} - V_0)^*(N_a{*}N_d)/(N_a+N_d)]^{1/2}\ [\ e^{-at} - e^{-bt}]. \qquad (3.18)$$

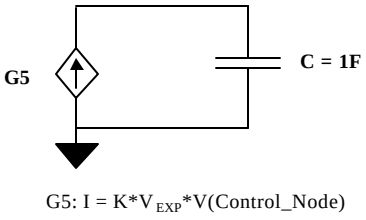
For the purposes of SPICE modeling, all terms other than the $(V_{node} - V_o)^{1/2}$ term are combined into a single constant K. The value used for $1/\alpha$ is 164 picoseconds, and for $1/\beta$ is 50 picoseconds from [26].



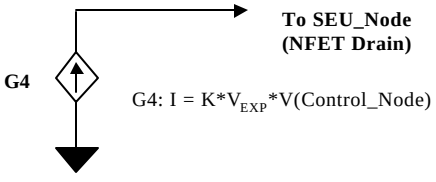
a. Double Exponential Voltage Source



b. Square Root Circuit



c. Charge Integration Circuit



d. Charge Injection Circuit

Figure 8. SET Injection Circuit (Low-Level)

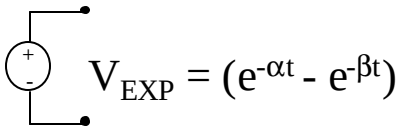
The circuits used in SPICE are shown in Figure 8. Figure 8a shows the voltage source that provides the double-exponential factor in the equation. Figure 8b shows the circuit that derives the $(V_{node} - V_o)^{1/2}$ term. This circuit is based on the Div and Sqrt circuits from [27]. The voltages from 8a and 8b are used as control voltages for the voltage-dependent current source in the Charge Injection Circuit of Figure 8d. These same control voltages drive the dependent current source G5 in Figure 8c to charge the 1F capacitor. At the end of the simulation, the voltage on this capacitor shows the total charge injected.

In high-level injection, the electric field of the junction has collapsed, and thus can no longer be modeled using equation 3.17. Instead, the electric field across the funnel is modeled as the field across a semiconductor bar with constant conductivity respect to the length, where the length is equal to the funnel length, L_f. Then,

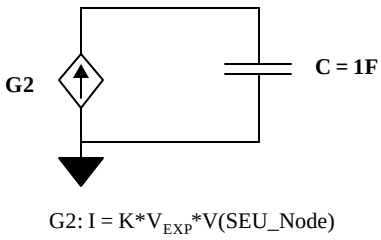
$$E = (V_{node} - V_{sub})/L_f. \qquad (3.19)$$

Substituting equation 3.18 into equation 3.7 and 3.8 gives:

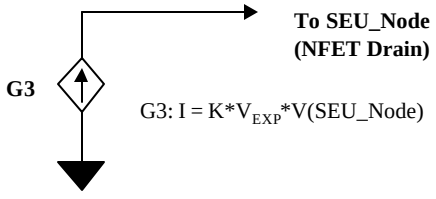
$$I(t) = [qmN\ (V_{node} - V_{sub})/L_f\ ]\ [\ e^{-at} - e^{-bt}]. \qquad (3.20)$$



a. Double Exponential Voltage Source



b. Charge Integration Circuit



c. Charge Injection Circuit

Figure 9. SET Injection Circuit (High-Level)

Figure 9 shows the SPICE circuit used to implement equation 3.20 for injection onto the drain of an NFET. The independent voltage source in Figure 9a provides the double-exponential term describing the carrier densities. The dependent current source, G3 is set equal to the product of a constant *K*, the double-exponential pulse from Figure 9a, and the SEU_Node voltage (V_{sub} = 0 for an NFET injection). *K* is constant for a single simulation run. It represents the product of qmN/L_f. Ions with different LETs are injected from one run to the next by changing *K*. This is equivalent to changing *N* from equation 3.4.

b. Conversion of Charge Collected (fC) to LET(MeV*cm^2/mg)

After the injection circuits described above have been used to simulate the SET, the charge collected on the 1F capacitor must be converted to LET in units of MeV*cm^2/mg. This is accomplished by assuming that each electron-hole pair created by the incident ion results in a charge equal to *q*, or 1.6e-19C. *q* is multiplied by equation 3.5 to give the total charge collected per unit length. By multiplying the constants in the equation, it can be determined that an ion with LET equal to 1 MeV*cm^2/mg will result in 10.35 fC/μm of collected charge. Then, combining this result with the funnel length, the simulated LET can be determined:

$$\text{LET(MeV*cm}^2\text{/mg)} = (\text{total injected charge in fC})/(L_f * 10.35). \qquad (3.21)$$

Equation 3.21 shows how critical the funnel length, L_f, is to the determination of the LET of the incident ion. In Dodd[28], 3-dimensional simulations were performed on a biased Si p+n junction with three different substrate doping levels. For doping levels similar to the KDLX processor modeled in Chapter IV, the simulations showed that a 100-MeV Fe ion strike (LET ~ 29.4 MeV*cm^2/mg or 0.306 pC/μm) will result in a total charge collection of 2.7 pC. The simulations also showed that the charge collection exhibited a breakpoint at 400 picoseconds. This is called the substrate breakpoint, and represents the breakpoint between funnel collection and diffusion collection. The diffusion collection is neglected because these simulations focus on the one to ten nanosecond timeframe, and diffusion does not add significantly to the charge collection during that time. At the end of the funnel collection timeframe, 1.2 pC had been

collected. This equates to a funnel length of 3.9 μm, which will be used in the modeling in Chapter IV.

C. SET CLOCK-EDGE EFFECTS MODELING

1. Objective

The objective of SET clock-edge effects modeling is to determine the probability that a transient pulse with amplitude = *a* and pulsewidth = *pw* will be latched into the memory element, or $P_{latch}(a, pw)$. The modeling focuses on determining the temporal relationship between the transient's arrival at the memory element and the edge of the control signal that latches it. The modeling also accounts for the effect of the amplitude of the transient pulse.

2. Underlying Theory

There are two timing parameters for memory elements that are key in modeling clock-edge effects: setup time (t_{su}) and hold time (t_h). Figure 10 shows a schematic for a pass-gate-type master-slave D-flip-flop and its associated timing diagram. The setup time is defined as the time data must be stable prior to the active edge of the clock (in this case, the positive edge). Smith [29] defines the hold time as the time data must be kept stable after the active edge of the clock. The setup time is determined by the time required for the input to propagate from D through inverter Inv1 to the input of Inv2. This propagation must occur before the passgate PG1 is turned off. The hold time (t_h) is determined by the minimum amount of time the data must be valid after PG1 has been turned off for the data to stabilize in the latch, which is created with inverters Inv2 and Inv3. The minimum pulsewidth ($t_{pw,min}$) required at the input D is given by:

$$t_{pw,min} = t_{su} + t_h. \qquad (3.21)$$

Thus, two criteria must be met for an SET to be latched: the pulsewidth must be greater than $t_{pw,min}$, and it must arrive at a time at least t_{su} prior to the active edge of the clock.

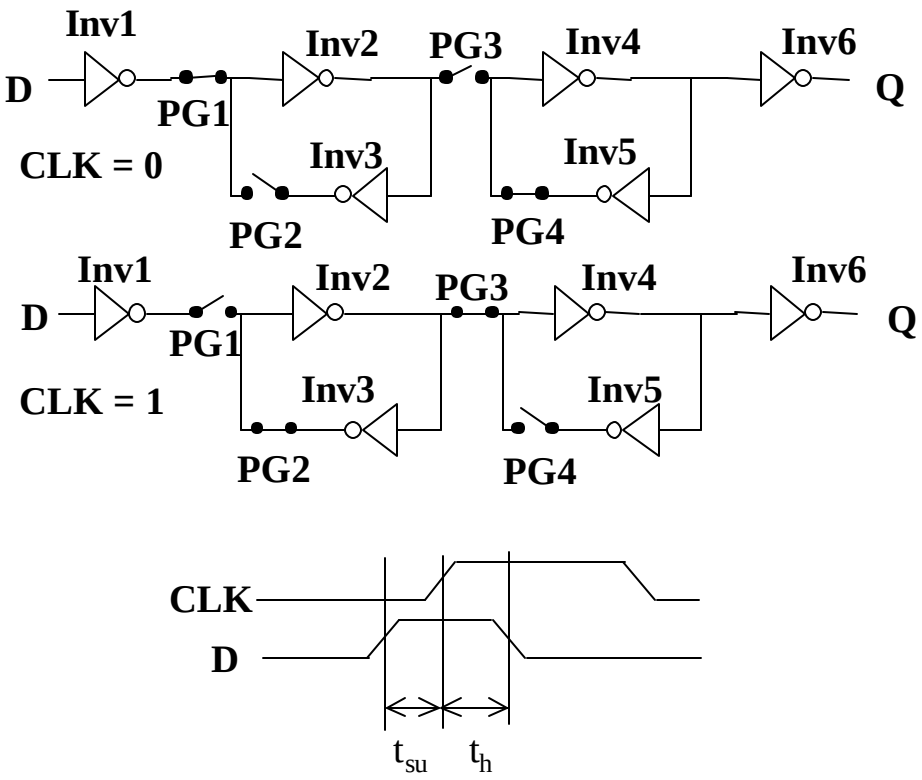


Figure 10. Setup and Hold Time

3. Previous Approaches

In Cha [16], SPICE simulations are used to determine the "latching window." A logic pulse is used as the input to a flip-flop. A logic pulse is defined as a pulse that makes the full rail-to-rail transition. The latching window for 0-1-0 and 1-0-1 pulses are determined as a function of the pulsewidth. The drawback with this approach is that the amplitude information of the SET is ignored.

In Buchner [30], the "window of vulnerability" is determined using laser pulses to inject transients. It is shown that the width of this window is a function of the energy of the laser pulse. This work is expanded on in Buchner[31]. It is shown, again using laser pulses, that there is a linear dependence on the probability of a transient being latched into the flip-flop.

4. Clock-Edge Effects Modeling Approach

The approach to modeling clock-edge effects uses SPICE to determine the latching window. However, unlike Cha[16], the pulse used is not a logic pulse. The

transient is injected in the circuit shown in Figure 11. It is injected one logic cell away from the input of the memory element at various times. The width of the transient is controlled by varying the amount of charge deposited. This approach maintains the appropriate transient-pulse shape going into the memory element.

To determine the latching window for a specific pulsewidth and amplitude of the transient pulse, the arrival time of the SET is varied to determine the maximum-setup time, $t_{su,\,max}$, and the minimum-setup time, $t_{su,min}$, for this particular pulse. These values are shown in Figure 12. The maximum setup time for a given amplitude and pulsewidth SET is the maximum time the SET can arrive prior to the active edge of the clock signal (CLK) and still be successfully latched. The signal D1 in Figure 12 shows an SET whose arrival time is equal to $t_{su,\,max}$. Similarly, the minimum setup time for a given amplitude and pulsewidth SET is the minimum time the SET can arrive prior to the active

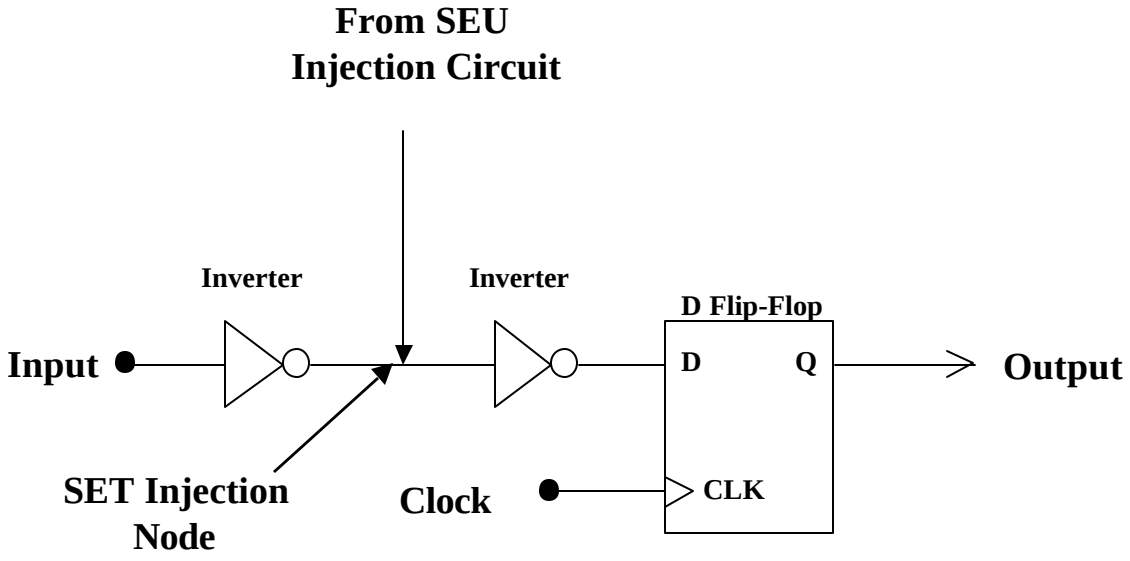


Figure 11. Clock-Edge Effects Simulation Circuit

edge of the clock signal (CLK) and still be successfully latched. Signal D2 in Figure 12 shows an SET whose arrival time is equal to $t_{su,\,min}$,. The latching window is then determined using the following equation:

$$t_{lw}(a, pw) = t_{su,\,max} - t_{su,min} \qquad (3.22)$$

Because the SET can only be latched once per clock cycle, the probability that the SET is latched is given by:

$$P_{latch}(a, pw) = t_{lw}(a, pw)/(\text{clock period}). \qquad (3.23)$$

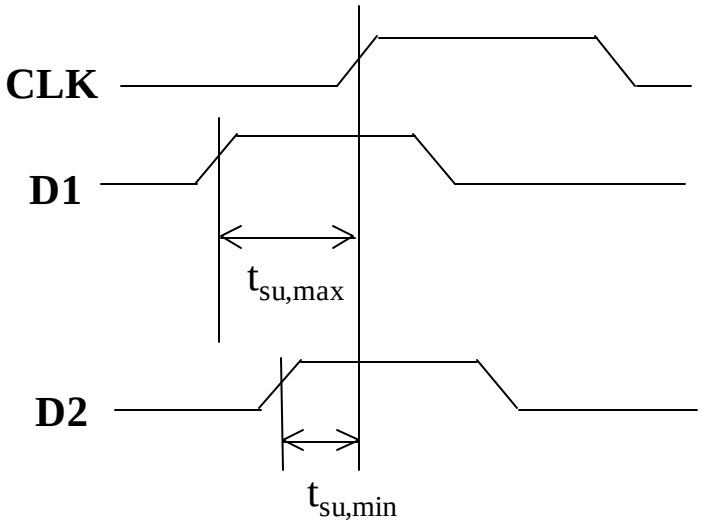


Figure 12. Latching Window Determination

D. SET ANALOG PROPAGATION MODELING

1. Objective

The purpose of analog propagation modeling is to determine what happens to the amplitude and pulsewidth of an SET as it propagates through a sensitized combinational-logic path. A sensitized combinational-logic path is defined as a path in which the propagation of the SET is not blocked by the other inputs to the logic in the path. For example, if an SET has propagated to input A of a 2-input AND gate, and input B is a logic "1," the logic path is sensitized. If input "B" had been a logic "0," the SET could not have passed through no matter what its amplitude and pulsewidth had been because the logic path was blocked (input B forces the output to logic "0").

2. Previous Approaches

Previous work in analog propagation has focused on SPICE simulations. The primary purpose of Cha [16] was to speed up SPICE-only transient simulation. SPICE is used to determine the resulting pulsewidth at the output of an inverter as a function of the quantity of charge injected and fan-out. This analog pulse is converted to a logic pulse using a threshold of Vdd/2. The logic pulse is then used for further simulations. While this approach succeeds in speeding up the simulation, it loses some fidelity by using the Vdd/2 threshold.

In Baze[32], the analog-simulation fidelity is maintained by performing a SPICE simulation of an SET propagating through a chain of inverters to the input of the flip-flop. This represents a high-fidelity approach, but also is very time consuming for a complex digital circuit.

3. SET Analog Propagation Approach

The approach used in this research is an improved version of the approach used in Cha [16]. The main difference with this approach is that the threshold is not arbitrarily chosen to be Vdd/2. Instead, the analog information is recorded, resulting in a higher fidelity simulation. An SET is injected into series of logic gates. The pulsewidth and amplitude are recorded as it propagates. From these values, a gate attenuation factor is determined in terms of pulsewidth and amplitude. Additionally, the propagation threshold is determined for a logic gate. The propagation threshold is the point at which the amplitude and pulsewidth of the SET is large enough such that it is not attenuated as it propagates. If an SET is at or above the propagation threshold, the attenuation factor is set to 1 (i.e., no attenuation).

E. SET LOGIC PROPAGATION MODELING

1. Objective

The objective of SET logic propagation modeling is to determine the probability that a sensitized combinational-logic path exists from the point of the SET generation to the input of the memory element. This probability is denoted as P_{scl}.

2. Previous Approaches

Baze [32] describes a method of determining the probability of error propagation in a complex circuit. The approach "uses a detailed cell level design description of a circuit to form a probabilistic mathematical model for static bit error propagation … The logic simulator performs a single simulation to obtain vector frequency distributions for all circuit cells and blocks … The propagation probability routine combines cell and block logic functions with state frequencies to calculate the numerical values of the propagation probabilities." This is a very thorough approach and requires a significant software effort.

Massengill [33] describes the SEUTool, which uses VHDL simulations to determine the likelihood that an SET will propagate to the input of a latch. These simulations are used to create an Error Probability Matrix, in which "each entry represents the probability, that, given a random SE[3] strike of strength Q_{coll} anywhere in the circuit of interest, that node N will cause an observable output error during clock cycle C." As with Baze [32], this approach is very thorough, but it requires a large VHDL simulation effort.

3. SET Logic Propagation Modeling Approach

The approach used in this dissertation is an improved version of the Baze approach. It is more efficient and reduces the computational complexity of the analysis. Logic is divided into two types: control logic and datapath logic. Control logic refers to logic that steers the flow of data through the possible datapaths. An example of control logic is a multiplexer that steers the flow of data from the output of the register file to the input of the arithmetic logic unit (ALU). Datapath logic is used in computations, but does not steer the flow of the data. An example is an OR gate used to create a fast adder in the ALU.

For control logic, the probability of logic propagation is assigned based on the how the datapath is steered. This is based on the functional mode of the system. For datapath logic, a random input is assumed. For example, for a 4-input AND gate, the probability that a transient will propagate through input "A" is 1/8. This is the likelihood that the other three inputs (B, C, and D) are equal to a logic "1."

F. SEU PROPAGATION MODELING

1. Objective

The purpose of SEU propagation modeling is to determine the probability that an SEU will propagate to the output and cause an output error. This addresses the $\varepsilon 1$ transitional probability on the SET-state-transition model. This transitional probability is very dependent on the functional mode of the digital system. A key aspect of SEU propagation modeling is the ability to express $\varepsilon 1$ as a mode-conditional probability.

[3] SE stands for Single Event.

2. Underlying Theory

When an SEU has occurred in a complex digital system, one of the internal memory elements is corrupted. From this point, four things can occur:

1. The SEU can be overwritten.
2. The SEU can remain.
3. The SEU can propagate internally creating multiple corrupted memory elements.
4. The SEU can propagate to the output.

3. Previous Approaches

There have been two primary approaches to determining which of these four possible outcomes will occur to the system. The first approach focuses on breaking the system into functional blocks (e.g., register file, ALU). Through testing, an attempt is made to determine the cross-section of the individual blocks. This is done by running different programs that stress different functional blocks. This is the approach used in Koga [14] and Asenek [34]. Asenek uses a "Duty Cycle Prediction Tool" to determine the duty cycle of each functional block. Heavy-ion testing and software simulations using an instruction-set simulator are then performed. The results from the testing and the simulations are the software-dependent upset rate. Each test program stresses a different functional block. Thus, the upset rate of a specific test program can be assigned to a specific functional block.

The second approach focuses on fault injection with hardware-description languages (HDL). This is the subject of Yount [35], Li [36], and Czeck[37]. In these papers, a fault is injected by changing the value of a single bit in an internal register during an HDL simulation. The output of the system is monitored to determine if any errors have propagated to the output.

4. SEU Modeling Approach

The SEU modeling approach used in this dissertation borrows from the two approaches described above. The duty-cycle approach is useful in that it provides a method of breaking down a complex digital system into functional blocks. The drawback is that more precise cross-section determination is desired. The second approach

described provides much greater fidelity of modeling, but the complexity of the simulation grows and becomes prohibitive as the complexity of the system grows.

The SEU modeling approach of this dissertation uses a combination of register-usage analysis and VHDL simulation. Register-usage analysis is used to reduce the complex digital system to a reasonable number of functional modes. For each possible mode, the registers that are necessary for proper execution within that mode are determined. These registers form the mode-dependent cross-section. For a processor, the complexity reduction is accomplished by considering each assembly language instruction as a unique mode. These instructions specify which registers within the functional blocks of the processor are being used. These instructions can be further broken down into the pipeline stages. For each pipeline stage of each instruction, the number of registers that must not be in error for proper instruction execution is determined. If a register is used, the number of clock cycles since it was last written is recorded. This provides a conditional probability of SEU propagation for each pipeline stage of each instruction.

In some cases, it is not apparent which bits of a register in a functional block add to the mode-dependent cross-section. In this case, fault injection in a VHDL simulation is used to provide additional insight. This is accomplished by injecting an error into each possible bit in the functional block and recording the resulting output errors. These results are then included in the higher-level register-usage analysis.

IV. MODELING AND SIMULATION

A. OBJECTIVE

The objective of this chapter is to verify the modeling methodology and approach described in Chapters II and III. This is accomplished by determining the previously defined transitional probabilities and using the SET-state-transition model (Figure 2) to determine the system-level upset rate for the KDLX processor, which is described in Appendix A. This processor was implemented in a custom layout with a standard-cell library and fabricated using the MOSIS prototyping service. As a result, the following information is available for modeling: parametric test results from the foundry run, an extracted transistor-level SPICE description, and a complete logic-gate-level VHDL description of the microarchitecture. This information is used for the SET generation modeling, SET propagation modeling, and SEU propagation modeling. The modeling results are combined to predict the system-level upset rate, which will be validated with measured upset rates in Chapter V.

B. SET GENERATION MODELING

1. Objective

The objective of the SET generation modeling is to determine the transitional probabilities $\beta 1$, $\beta 2$ and $\beta 3$. As discussed in Chapter 3, $\beta 1$ will be described as cross-section versus LET curves. $\beta 2$ and $\beta 3$ will be described as cross-section versus LET, resulting pulsewidth, and resulting amplitude tables.

2. Determination of Key Parameters

The first step in SET Generation modeling is to determine key parameters from the MOSIS parametric test results and the extracted layout information. These parameters are necessary to determine the sensitive cross-section versus LET curves from the SPICE modeling. Specifically, these parameters are the depletion width, contact potential, doping levels, and low-level/high-level-injection crossover point.

Table 2 shows the parameters that are given in the MOSIS parametric test results for the wafer run used in the fabrication of the KDLX device. Table 3 shows the derived parameters. The doping levels of the n-channel and p-channel devices were

Parameter	Value
N-Channel Electron Mobility	400.02 $cm^2/(V*s)$
P-Channel Hole Mobility	136.52 $cm^2/(V*s)$
Measured N-Channel to Substrate Area Capacitance	494 $aF/\mu m^2$
Measured P-Channel to N-well Area Capacitance	943 $aF/\mu m^2$

Table 2. MOSIS Parametric Test Results [38]

determined using the measured low-field mobility of the n-channel and p-channel devices with the mobility-versus-doping-level charts from Jacobini [39]. The n-well and substrate doping levels are determined by using the channel doping levels and the measured area capacitance values for measured n-channel-to-substrate area capacitance and the measured p-channel-to-n-well area capacitance with the equation for junction capacitance from Streetman[40]:

$$C_j = eA\{q*N_d*N_a/[(V_0-V)(N_d+N_a)]\}^{1/2}, \qquad (4.1)$$

where V is the voltage applied during the parametric test (in Volts).

Parameter	Value
N-Channel Doping	5e17 $donors/cm^3$
P-Channel Doping	5e17 $acceptors/cm^3$
N-channel to Substrate Contact Potential (Φ)	0.82 Volts
P-channel to N-well Contact Potential (Φ)	0.86 Volts
Calculated Substrate Doping Level	2.51e16 $acceptors/cm^3$
Derived Substrate Hole Mobility	325 $cm^2/(V*s)$
Calculated N-well Doping Level	1.12e17 $donors/cm^3$
Derived N-well Electron Mobility	780 $cm^2/(V*s)$
NFET Depletion Depth (W)	0.474 μm
PFET Depletion Depth (W)	0.244 μm
Low-Level/High-Level Injection Crossover-Point LET	0.245 $MeV*cm^2/mg$

Table 3. Derived Parameters

The low-level/high-level injection crossover-point was calculated to determine which of the SET injection circuits to use (Figure 8 or Figure 9). Table 3 shows that the crossover LET is 0.245 $MeV*cm^2/mg$. When the corresponding value of charge (~ 10

femto-coulombs) was injected on the standard-cell inverter in a SPICE simulation, the result was nearly imperceptible – only several millivolts. This is because the quantity of charge injected was too small. To see any effect, the simulated LET must be increased significantly above the crossover LET. Therefore, high-level injection was modeled for all the SET-generation simulations.

3. Determination of Transitional Probability $\beta 1$

The transitional probability $\beta 1$ is the likelihood that an SET occurs on a transistor within a memory element with enough energy to directly cause an SEU. As described in Chapter III, this probability will be modeled as a cross-section versus LET curve. The only memory element in the KDLX design is the D-Flip-Flop-with-asynchronous-clear (DFFC) standard cell. The schematic for the DFFC is shown in Figure 13. Determining the cross-sections and LETs for $\beta 1$ requires four input cases to be simulated: Clk=0, Data=0; Clk=0, Data=1; Clk=1, Data=0; Clk=1, Data=1. The ClB Input was set to logic "1" to simulate normal operation. For each input case, the sensitive transistors were determined. For each sensitive transistor, several SETs were injected using the high-level-injection circuit. The amount of charge deposited from the SETs was varied until the minimum charge necessary to cause an SEU was determined. Table 4 shows this minimum charge (also known as the critical charge) and corresponding LET required to cause an upset for each sensitive transistor for each input case. It also shows the cross-section area of the sensitive transistors. Figure 14 shows the cross-section-versus-LET curve for a single DFFC standard cell.

Table 4 and Figure 14 show that the onset LET for the DFFC should occur at 8.4 MeV*cm^2/mg, which corresponds to 339 fC deposited on the drain of PFET T17. The effective cross-section of T17 is 2.31 µm. As the LET is increased, the critical charge of all the transistors is reached. This occurs when the LET is equal to 23 MeV*cm^2/mg. At this point, the effective-saturated cross-section of the DFFC is 33.66 µm.

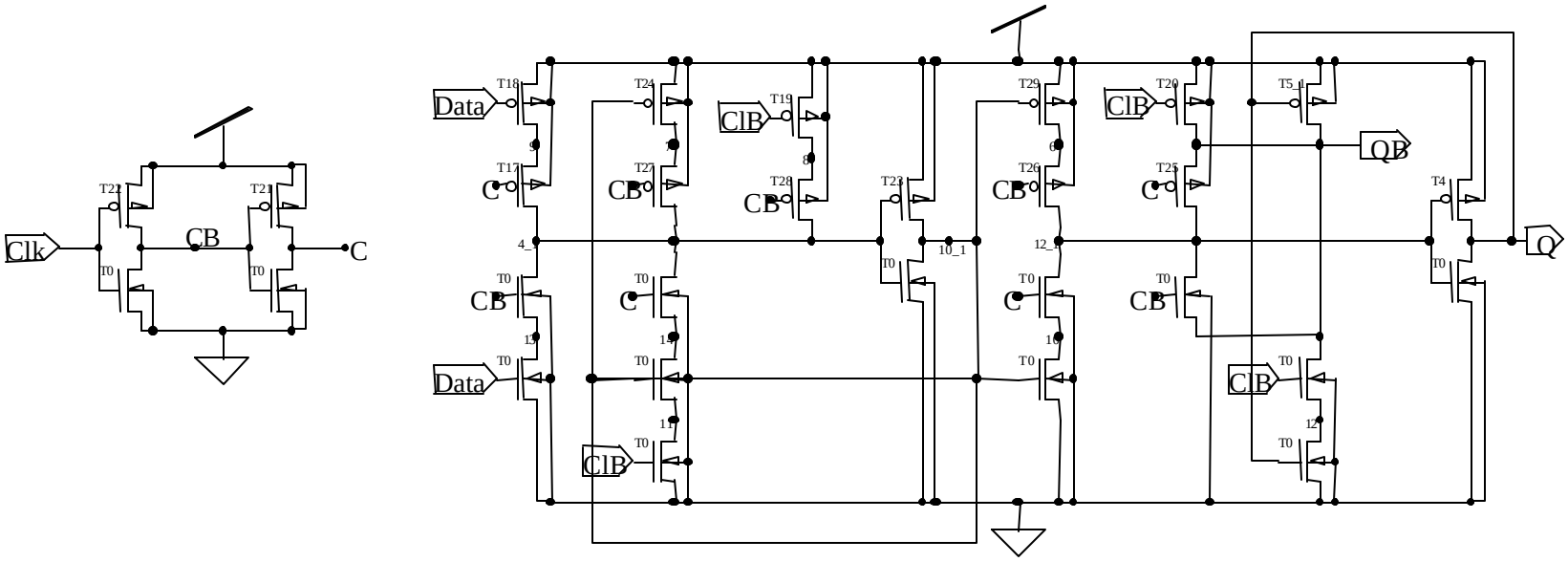


Figure 13. DFFC D-Flip-Flop Schematic (after [41])

Input State	Sensitive Transistor	Cross-Section (μm^2)	Effective Cross-Section (μm^2)	Critical Charge (fC)	LET (MeV*cm^2/mg)
Data =0 Clk =0	T15 NFET	8.72	2.18	439	10.9
Data =0 Clk =0	T3_1 NFET	12.46	3.11	930	23.0
Data =0 Clk =0	T4 PFET	10.06	2.51	834	20.7
Data =0 Clk =1	T13 NFET	8.72	2.18	376	9.3
Data =0 Clk =1	T12 NFET	4.72	1.18	421	10.4
Data =0 Clk =1	T23 PFET	10.06	2.51	619	15.3
Data =1 Clk =0	T2_1 NFET	28.19	7.05	588	14.6
Data =1 Clk =0	T26 PFET	7.56	1.89	373	9.2
Data =1 Clk =0	T20 PFET	11.19	2.80	717	17.8
Data =1 Clk =0	T5_1 PFET	16.55	4.14	717	17.8
Data =1 Clk =1	T11 NFET	11.03	2.76	473	11.7
Data =1 Clk =1	T17 PFET	9.24	2.31	339	8.4

Table 4. DFFC Sensitive Transistor Critical Charge, LET, and Cross-Section

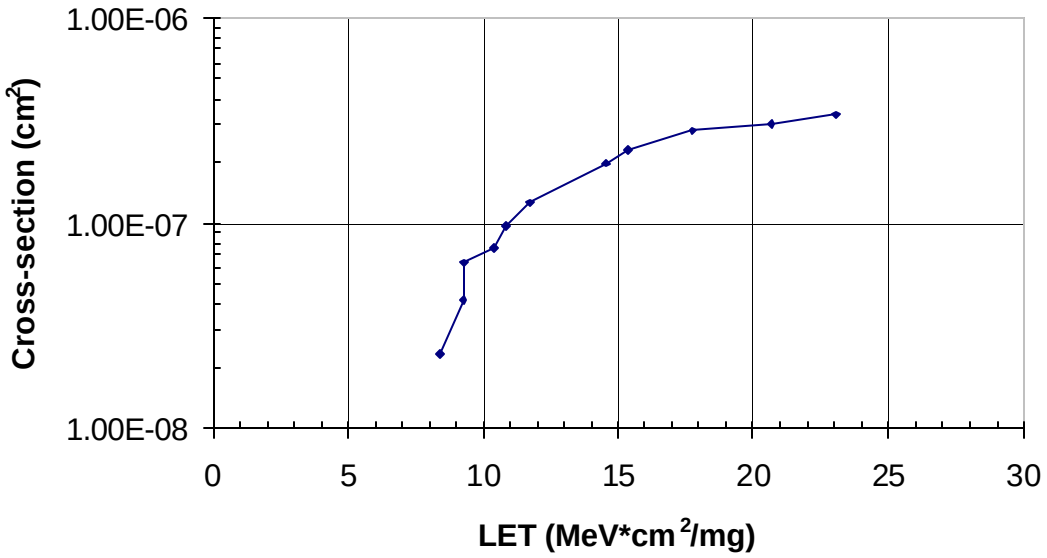


Figure 14. DFFC Cross-Section Versus LET Curve

4. Determination of Transitional Probability b2

The transitional probability β2 is the likelihood that an SET occurs on a combinational-logic gate. As defined in Chapter III, β2 will be listed as a cross-section, LET, and resulting amplitude and resulting pulsewidth. Thus, to determine β2, the injection circuit is used to inject an SET into the sensitive regions of each of the standard cells. The charge injected is converted to LET, and the output pulsewidth and amplitude are recorded. In Section C5 of this chapter, these resulting output pulsewidths and amplitudes are coupled with the results of the SET analog propagation, logic propagation and clock-edge effects modeling to determine the probability that the SET will become latched. The standard-cell inverter is described as an example.

Figure 15 shows the schematic of the inverter and the test circuit. The output of the inverter is connected to the input of another inverter. This insures proper output loading. If the input is equal to logic '0', the NFET is sensitive. If the input is logic '1', the PFET is sensitive. Figure 16a shows the current waveform from the high-level injection circuit and Figure 16b, the resulting voltage on the injection node. Figure 17

shows this resulting voltage for various LETs for the injected pulse. This figure shows that an LET of approximately 13.89 MeV*cm^2/mg is necessary for the SET to make the full voltage swing. As the LET is increased beyond 13.89 MeV*cm^2/mg, the pulsewidth of the SET increases. Table 5 shows the cross-section, LET, resulting amplitude and resulting pulsewidth for the inverter.

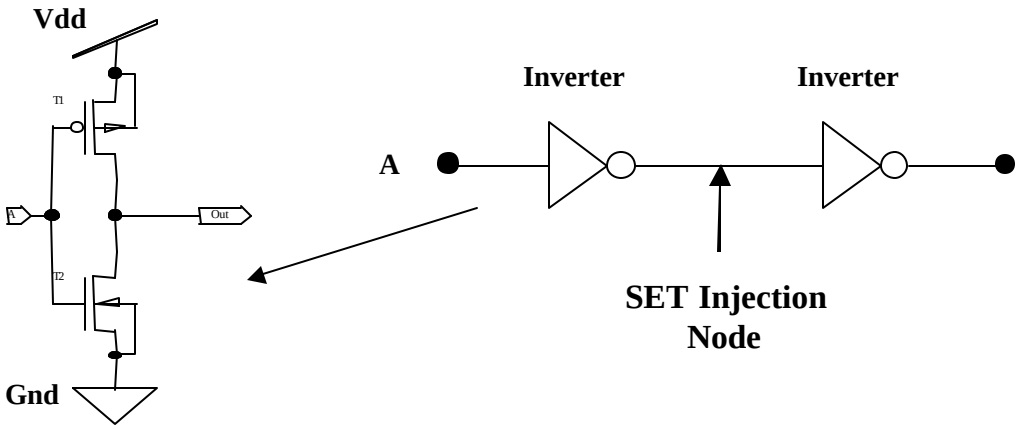


Figure 15. Inverter Standard-Cell Schematic and Test Circuit

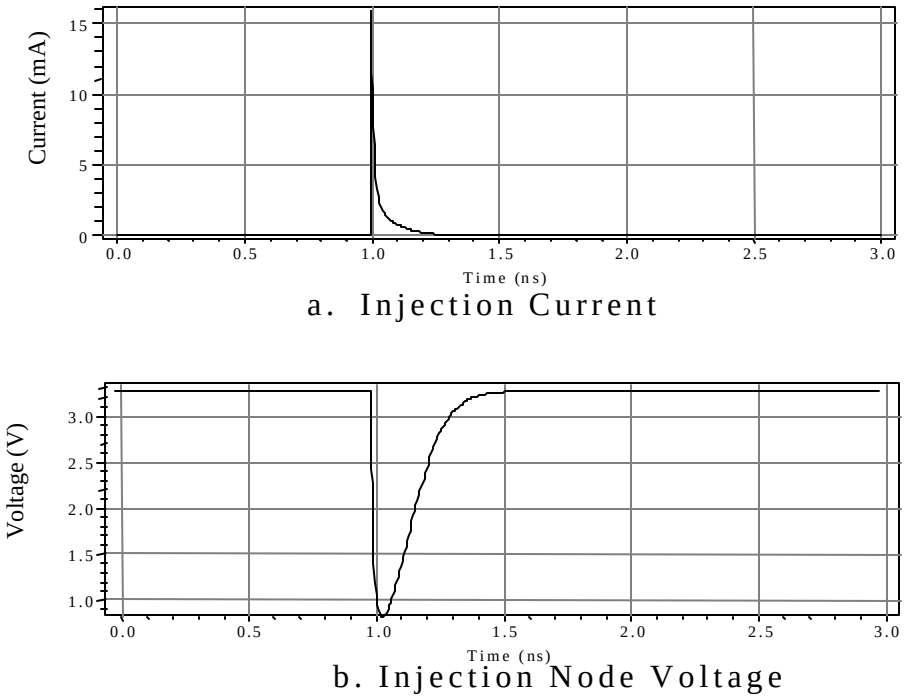


Figure 16. Injection Current and Node Voltage

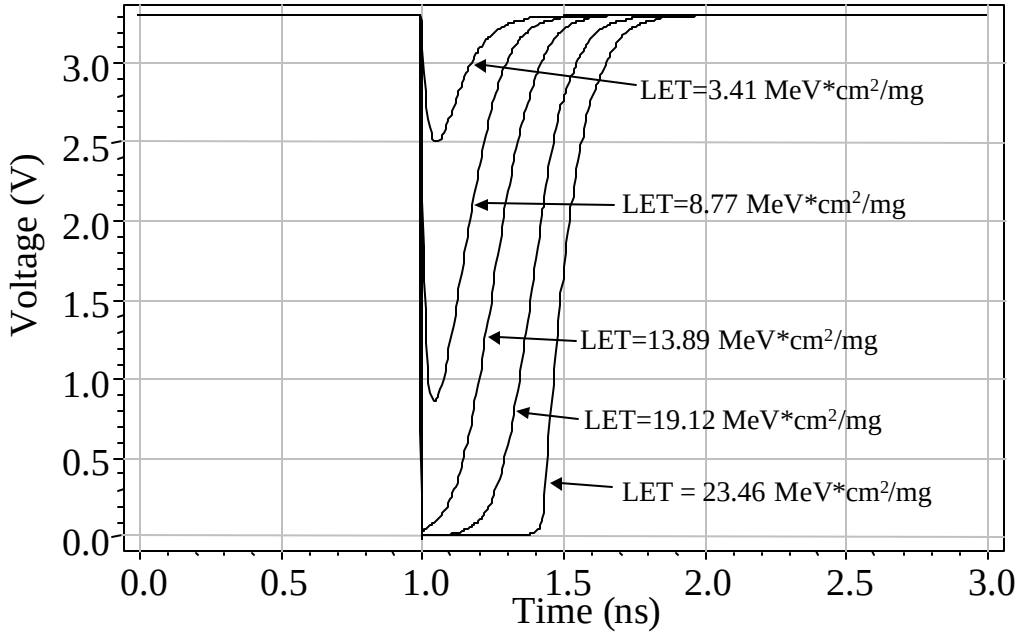


Figure 17. SET Pulse Shape Versus LET

Input State	Sensitive Transistor	Charge (fC)	LET (MeV*cm^2/ mg)	Cross-Section (μm^2)	Effective Cross-Section (μm^2)	Output Amplitude (Volts)	Output Pulse-Width (ps)
A=0	T2 (NFET)	137.6	3.41	41.59	20.80	-0.83	140
A=0	T2 (NFET)	353.9	8.77	41.59	20.80	-2.44	180
A=0	T2 (NFET)	423.7	10.50	41.59	20.80	-3.0	190
A=0	T2 (NFET)	560.8	13.89	41.59	20.80	-3.3	260
A=0	T2 (NFET)	771.9	19.12	41.59	20.80	-3.3	400
A=0	T2 (NFET)	880.3	21.81	41.59	20.80	-3.3	470
A=0	T2 (NFET)	947	23.46	41.59	20.80	-3.3	490
A=1	T1 (PFET)	140	3.47	29.98	14.99	0.474	90
A=1	T1 (PFET)	477.3	11.82	29.98	14.99	1.91	100
A=1	T1 (PFET)	621	15.38	29.98	14.99	2.69	120
A=1	T1 (PFET)	907.6	22.48	29.98	14.99	3.2	200
A=1	T1 (PFET)	1260	31.22	29.98	14.99	3.3	300
A=1	T1 (PFET)	1530	37.90	29.98	14.99	3.3	390
A=1	T1 (PFET)	1670	41.37	29.98	14.99	3.3	440
A=1	T1 (PFET)	1720	42.61	29.98	14.99	3.3	460

Table 5. Cross-Section and LET for Standard-Cell Inverter

5. Determination of Transitional Probability b3

The transitional probability β3 is the likelihood that an SET occurs on an output driver. This simulation is similar to modeling the inverter to determine β2, except the output driver is connected to an output pad plus an 8 pF capacitor. The 8 pF capacitor is the input capacitance of a Xilinx XCV300 Field Programmable Gate Array (FPGA) [42], which is the device connected to the KDLX in the test system. The results of the simulation are shown in Table 6.

Input State	Sensitive Transistor	Charge (fC)	LET (MeV*cm^2/mg)	Cross-Section (mm^2)	Effective Cross-Section (mm^2)	Output Amplitude (Volts)	Output Pulse-Width (ps)
A=0	T1 (PFET)	12960	347.83	84.47	42.24	1.13	800
A=0	T1 (PFET)	25730	690.55	84.47	42.24	2.37	1050
A=0	T1 (PFET)	30530	819.38	84.47	42.24	2.87	1210
A=0	T1 (PFET)	33060	887.28	84.47	42.24	3.08	1280
A=0	T1 (PFET)	33860	908.75	84.47	42.24	3.18	1290
A=0	T2 (NFET)	12830	344.34	127.04	63.52	-1.27	1970
A=0	T2 (NFET)	25250	677.67	127.04	63.52	-2.47	2470
A=0	T2 (NFET)	29870	801.66	41.59	20.80	-2.93	2690
A=0	T2 (NFET)	31600	848.09	41.59	20.80	-3.14	2750
A=0	T2 (NFET)	32280	866.34	41.59	20.80	-3.22	2820

Table 6. SET on Output Driver

Comparing the results in Table 6 to the results Table 5 shows that an SET on an output driver requires a much greater quantity of charge to reach a given amplitude than an SET on an internal node. This is a direct result of the larger capacitance of the output device relative to the capacitance of an internal node. As shown in the table, an ion incident upon the PFET requires an LET greater than 347 MeV*cm^2/mg to result in a transient with an amplitude greater than 1.13 Volts. Similarly, an ion incident upon the NFET requires an LET greater 343 MeV*cm^2/mg to cause a transient with an amplitude

greater than 1.27 Volts. From Ziegler[43], the largest linear energy transfer in silicon from a heavy ion is ~ 120 MeV*cm^2/mg. Since 343 >> 120, $\beta 3$ can be set to 0, and the output drivers of the KDLX are modeled as not susceptible to SETs.

C. SET PROPAGATION MODELING

1. Objective

The objective of SET propagation modeling is to determine the propagation transitional probabilities $\delta 1$, $\delta 2$, and $\varepsilon 2$. This can be broken down into three parts: SET analog propagation, SET logic propagation, and clock-edge effects. The results of these simulations are coupled with the results of the SET generation modeling.

2. SET Analog Propagation Modeling

a. Objective

The objective of the SET analog propagation modeling is to determine what happens to the amplitude and pulsewidth of an SET as it propagates through the logic gates used in the KDLX. The results are used to determine the probability of analog propagation through a sensitized logic path.

b. Modeling Configuration

The circuit shown in Figure 18 is used to model the analog propagation through the standard cell inverter. Using the SET injection circuit, the transient is injected at the node named SET_Node. The injection circuit is used (as opposed to a logic pulse) to insure that the rise and fall times are consistent with an SET. The propagating transients are observed at Prop_Node1, Prop_Node2, Prop_Node3, Prop_Node4 and Prop_Node5.

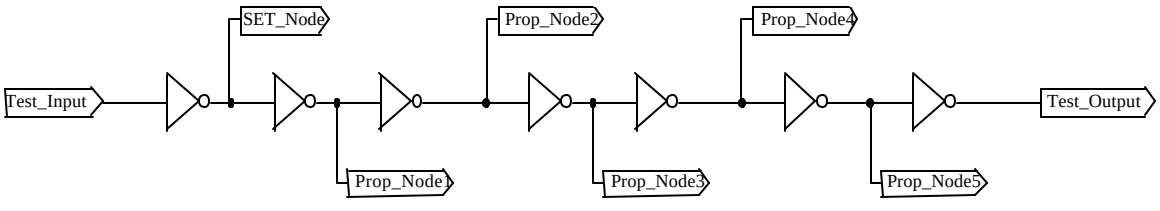


Figure 18. Inverter Propagation Circuit

c. Modeling Results

Figure 19 shows the propagation of a small transient. As it passes through each inverter, the transient attenuates significantly. In fact, by the time it has propagated to Prop_Node4, the amplitude is less than 100 mV. Figure 20 shows the propagation of a slightly larger transient. In this case, attenuation is also occurring through each inverter, but it is not as rapid as in Figure 19. In both cases, there is not enough energy in the transient to propagate without attenuation. In contrast, Figure 21 shows the propagation of a transient that does not attenuate at all as it propagates. At some point between the size of the transients in Figures 20 and 21, there is a threshold above which transients will propagate without attenuation. This is defined as the propagation threshold.

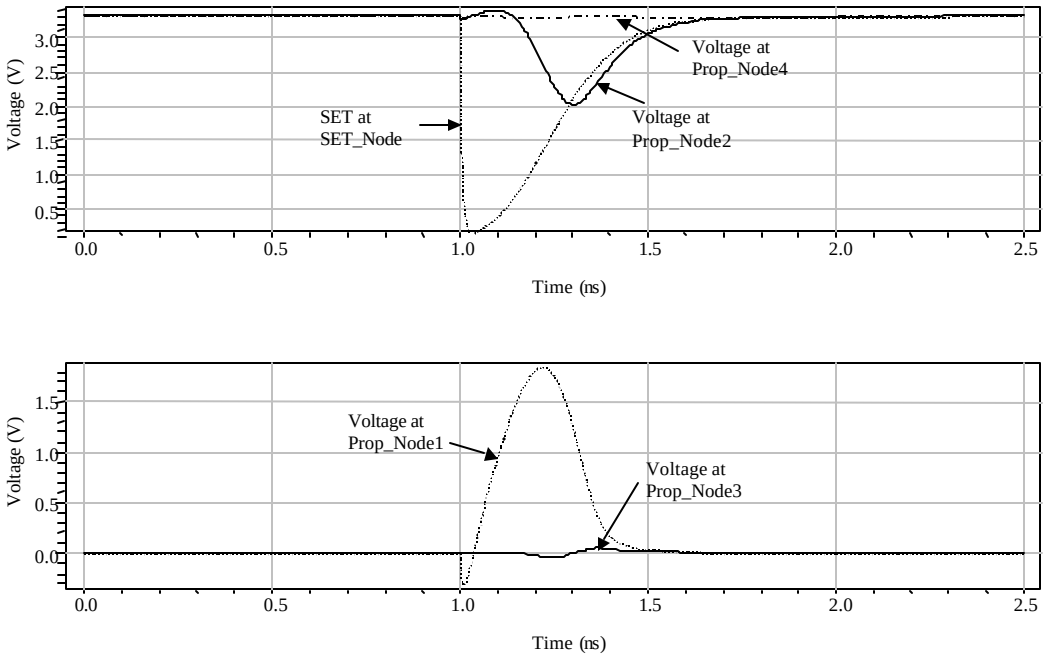


Figure 19. Propagation of Small SET

The propagation threshold is an important concept in SET propagation modeling. This is because the probability of SET analog propagation through a path of logic gates can be set to 1 if the SET is above the propagation threshold. Furthermore, if the latching threshold (to be determined in the Clock-Edge Effects section) is greater than the propagation threshold, and the SET meets or exceeds the latching threshold, then it

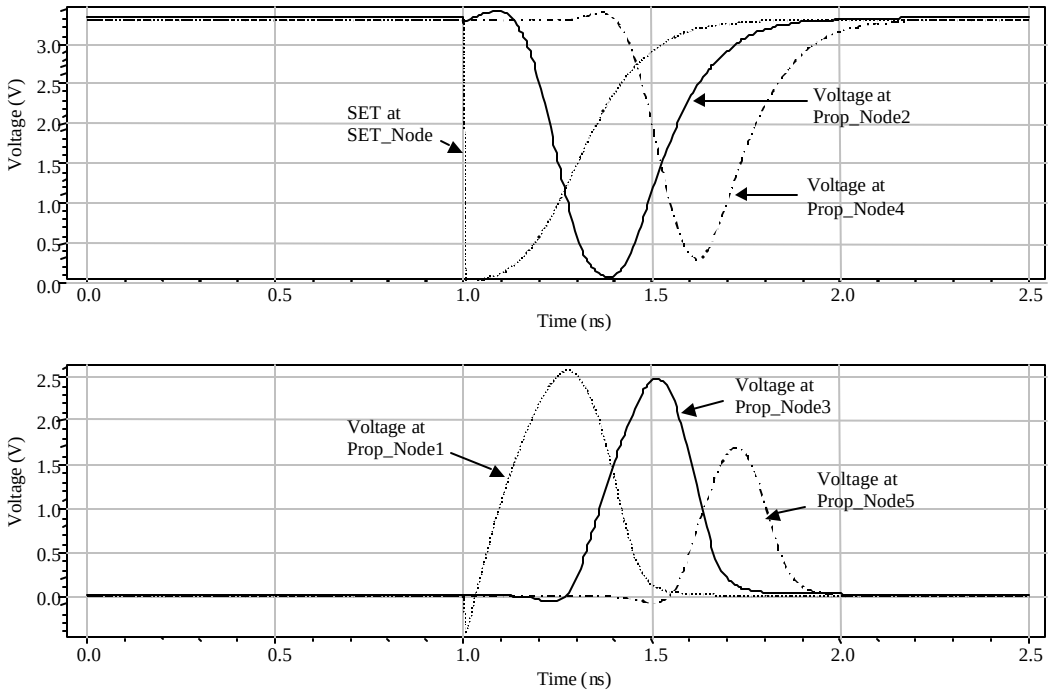


Figure 20. Propagation of Medium SET

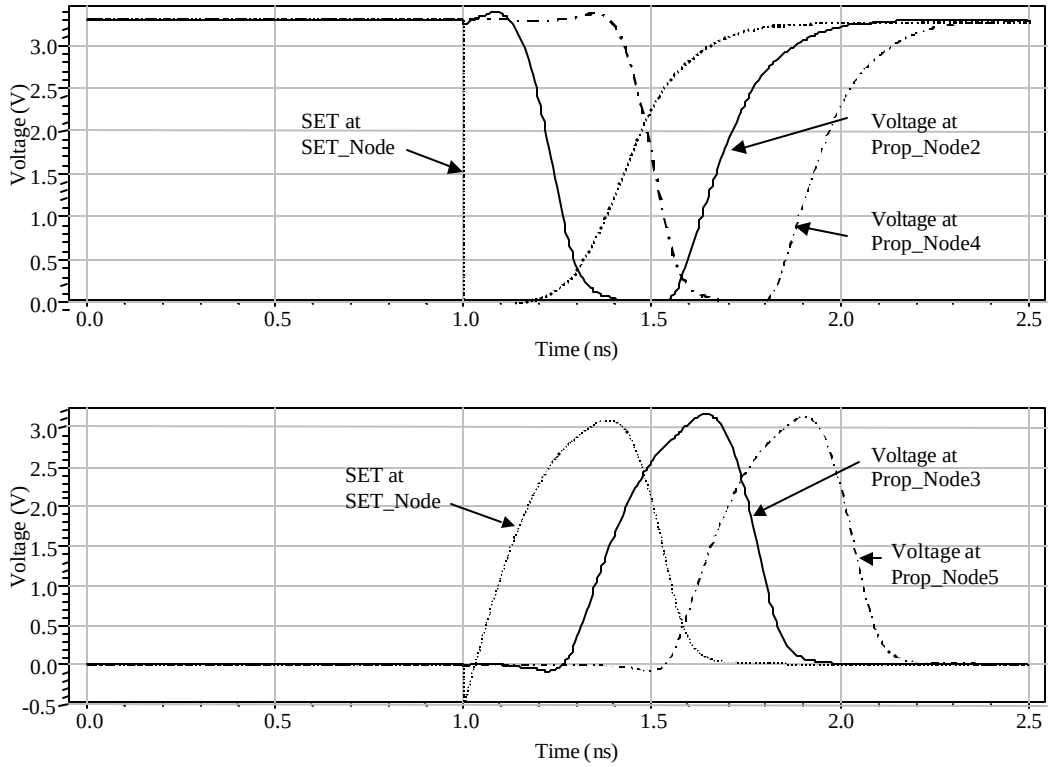


Figure 21. Propagation of Large SET

must also exceed the propagation threshold, and the probability of SET analog propagation can be set to 1.

To determine the propagation threshold, the simulation is run with multiple SETs injected into the circuit. The amplitude and pulsewidth are measured at each node. Table 7 shows the results of these simulations. The propagation threshold for a 0-1-0 SET pulse is approximately an amplitude of 3 V and a pulsewidth of 400 picoseconds. The propagation threshold for at 1-0-1 SET pulse is an amplitude of 3.3V and a pulsewidth of 460 picoseconds. These propagation simulations are repeated for the other standard cells, and the results are shown in Appendix B.

SET_Node		**Prop_Node1**		**Prop_Node2**		**Prop_Node3**		**Prop_Node4**	
Amp (V)	**PW (ps)**	**Amp (V)**	**PW (ps)**	**Amp (V)**	**PW (ps)**	**Amp (V)**	**PW (ps)**	**Amp (V)**	**PW (ps)**
-2.9	240	1.36	200	NA	NA	NA	NA	NA	NA
-3.25	290	2.2	260	-2.58	250	1	140	-.08	180
-3.27	300	2.45	280	-3.11	280	2.16	210	-2.05	220
-3.28	330	2.6	300	-3.26	310	2.59	250	-3.18	270
-3.3	400	2.96	380	-3.3	400	3.06	340	-3.3	390
-3.3	450	3.15	410	-3.3	460	3.22	400	-3.3	460

Amp = Amplitude in Volts (V), PW = pulsewidth in picoseconds (ps)

Table 7. SET Propagation - Inverter

3. SET Logic Propagation Modeling

SET logic propagation modeling determines the probability that an SET will propagate through the logic gate, given that the amplitude and pulsewidth are large enough for analog propagation. Table 8 shows the probability of logic propagation for each of the standard-cell logic gates used in the KDLX design. For multiple-input logic gates that are not instruction-dependent, the inputs are modeled as random. For the Mux2, the probability is modeled as being instruction-dependent. This is because the

Mux2 is used throughout the KDLX to direct the data path as a function of the instruction, whereas the other multiple logic gates have inputs that are not direct functions of the instruction. This is critical because it causes δ1 to be instruction-dependent (if there is a Mux2 in the datapath). Additionally, the gates that are used in the decoding logic of the pipeline are modeled as instruction-dependent.

Standard Cell	Probability of Logic Propagation
Inv	1
Buf4	1
Nand2	0.5 (Non-Pipeline) Instruction-Dependent (Pipeline)
Nand3	0.25
Nand4	0.125 (Non-Pipeline) Instruction Dependent (Pipeline)
Nor2	0.5 (Non-Pipeline) Instruction-Dependent (Pipeline)
Nor3	0.25
Nor4	0.125 (Non-Pipeline) Instruction-Dependent (Pipeline)
Xor2	1
Mux2	Instruction-Dependent

Table 8. Probability of Logic Propagation

4. Clock-Edge Effects Modeling

a. Objective

The objective of the clock-edge effects modeling is to determine the latching window as a function of the amplitude and pulsewidth of the SET, denoted as $t_{lw}(a, pw)$. The latching window will be used to determine the probability that the SET is latched as a function of the amplitude and pulsewidth: $P_{latch}(a, pw)$. This probability will be combined with the analog propagation and logic propagation modeling results to determine the transitional probability δ1 in the following section.

b. Modeling Configuration

The circuit in Figure 22 is the simulation circuit used to model the clock-edge effects. The SET injection circuit is used to inject the SET to the node named "SET_Node." As with the SET analog propagation modeling, the SET injection circuit is

used to insure that only amplitudes and pulsewidths that can result from an SET are used. The resulting amplitude and pulsewidth as the SET propagates to the node named "DATA" is recorded. To determine the latching window for an SET with a specified amplitude and pulsewidth, the time the SET is injected onto SET_Node is varied to determine the minimum setup time ($t_{su,min}$) and maximum setup time ($t_{su,\ max}$) for the specified amplitude and pulsewidth. The latching window for the specified amplitude and pulsewidth is determined using equation 3.22:

$$t_{lw}(a, pw) = t_{su,\ max} - t_{su,min.} \qquad (3.22)$$

This is process is repeated for other amplitude and pulsewidth combinations to determine the latching window as a function of amplitude and pulsewidth.

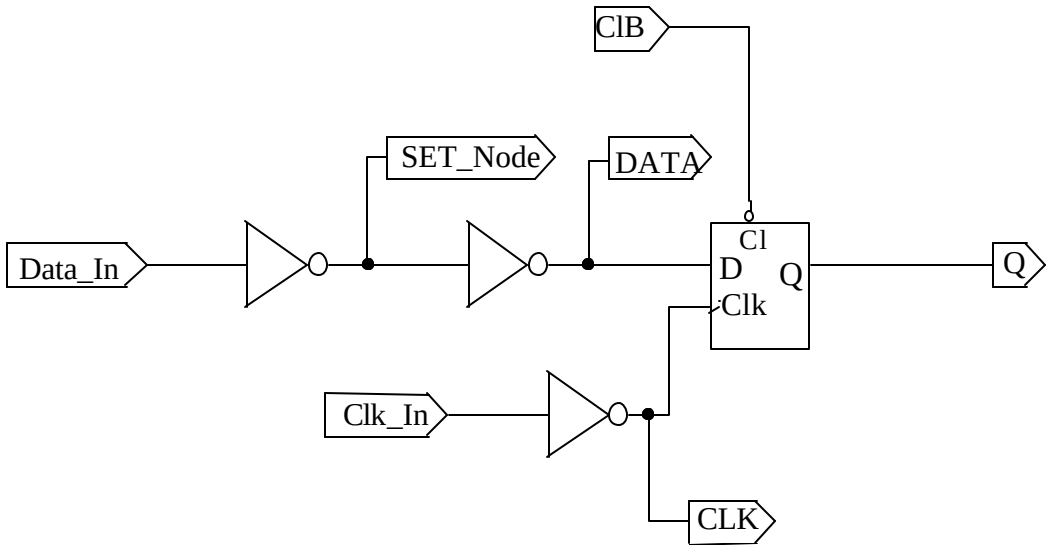


Figure 22. Clock-Edge Effects Modeling Circuit

c. Modeling Results

Table 9 shows the latching window as a function of the SET amplitude and pulsewidth. If the SET pulse arrives during the latching window and has sufficient energy, it will be latched. The probability of the SET pulse being latched is given by the equation below:

$$\mathbf{P_{latch} = (latching\ window)/(clock\ period)} \qquad (4.2)$$

SET Amplitude (V)	SET Pulse-width (ps)	Latching Window (ps)	P_{latch} (1/MHz)	P_{latch} @ 625 KHz	P_{latch} @ 5 MHz
-3.3	480	60	6.00E-05	3.75E-05	3.00E-04
-3.3	490	80	8.00E-05	5.00E-05	4.00E-04
-3.3	500	180	1.80E-04	1.13E-04	9.00E-04
-3.3	510	190	1.90E-04	1.19E-04	9.50E-04
-3.3	520	230	2.30E-04	1.44E-04	1.15E-03
-3.3	530	270	2.70E-04	1.69E-04	1.35E-03
-3.3	550	340	3.40E-04	2.13E-04	1.70E-03
-3.3	560	360	3.60E-04	2.25E-04	1.80E-03
3.3	510	70	7.00E-05	4.38E-05	3.50E-04
3.3	520	140	1.40E-04	8.75E-05	7.00E-04
3.3	560	210	2.10E-04	1.31E-04	1.05E-03
3.3	580	240	2.40E-04	1.50E-04	1.20E-03
3.3	600	280	2.80E-04	1.75E-04	1.40E-03
3.3	640	330	3.30E-04	2.06E-04	1.65E-03
3.3	670	370	3.70E-04	2.31E-04	1.85E-03
3.3	690	400	4.00E-04	2.50E-04	2.00E-03

Table 9. Clock-Edge Effects Modeling Results

Because of the relationship between this probability and the clock frequency, P_{latch} is listed in units of 1/MHz and also as a probability at two specified clock frequencies: 625 KHz and 5 MHz.

The table shows that there is an SET latching threshold. For a 1-0-1 transition, the SET must have an amplitude of 0V (full –3.3V transition) and a pulsewidth of 480 picoseconds. For a 0-1-0 transition, the threshold is an amplitude of 3.3V and a pulsewidth of 510 picoseconds. Below these thresholds, the SET will not be latched. In comparison, the propagation threshold (from Table 7) requires a 400 picosecond SET pulsewidth. A close look at the SET propagating within the flip-flop shows the reason the latching threshold is higher than the propagation threshold. Node 4_1 of the DFFC schematic shown in Figure 13 is the critical node in the determination of the latching threshold. Figure 23 shows this node voltage for an SET that is slightly above threshold. Figure 24 shows this voltage for an SET that is slightly below threshold. In both cases, the SET arrives at the DATA input. With the clock low, transistor T13 is turned on. The transient is attenuated as it passes to Node 4_1. This is because the on-resistance of T13

coupled with the capacitance at node 4_1 form a low-pass filter that removes the high frequency components of the transients. Transients with wider pulsewidths have more energy at lower frequencies and more energy is passed through the low-pass filter. In Figure 23, the transient has enough energy (which is to the product of charge and voltage) after this attenuation to keep the voltage at Node 4_1 at logic "0" when the rising edge of the clock occurs. In Figure 24, the transient is able to pass some energy to Node 4_1. However, not enough energy is passed through for the voltage at Node 4_1 to be latched in. Thus, the smaller transient is not latched.

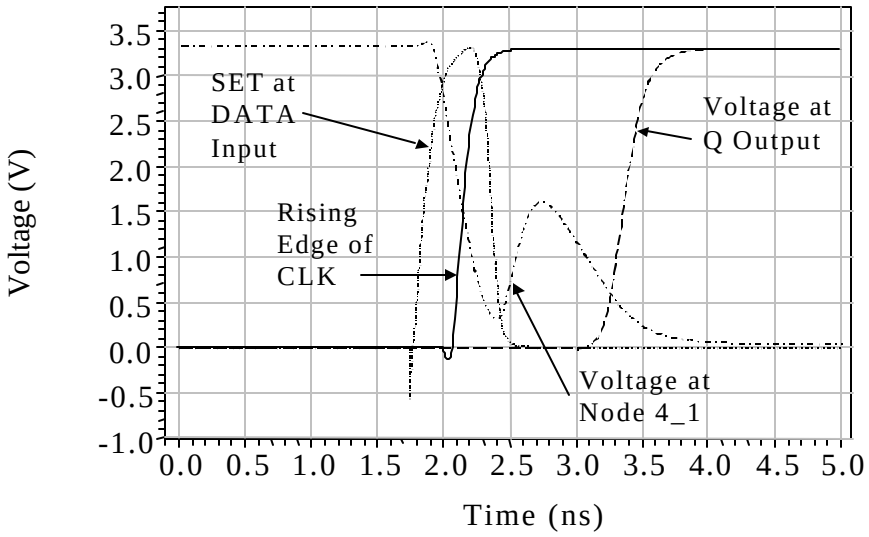


Figure 23. SET Above Latching Threshold

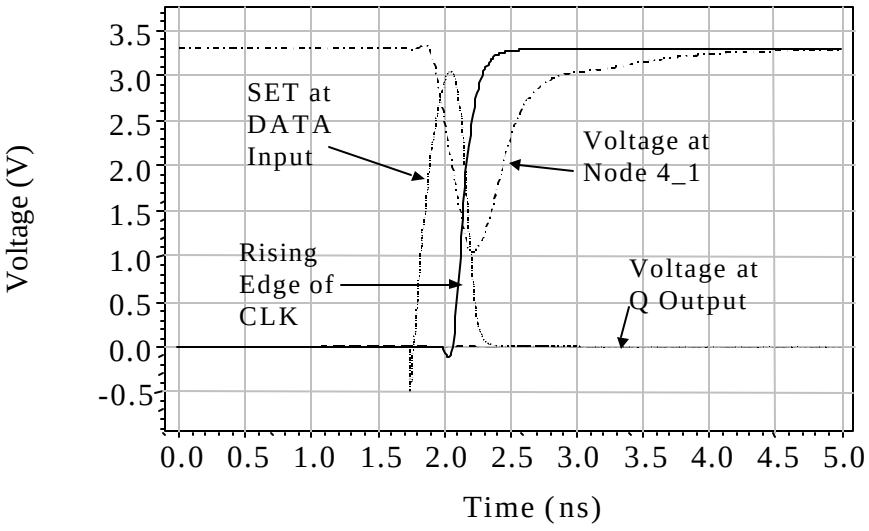


Figure 24. SET Below Latching Threshold

Because the latching threshold is greater than the propagation threshold, the latching threshold defines the minimum amplitude and pulsewidth for an SET in logic to be latched and become an SEU. This simplifies the determination of $\delta 1$, because if the SET meets the latching threshold requirements, the probability of analog propagation is equal to one. If an SET does not meet the latching threshold requirements, $\delta 1$ is set to zero because it will not be latched.

5. Determination of the Transitional Probability $\delta 1$

From Figure 2, $\delta 1$ is the probability that an SET will propagate from the sensitive region of a logic gate where generation occurred to the input of the memory element AND be latched in. Thus, $\delta 1$ is the product of the P_{latcch} (a, pw) * P_{scl}* P_{ap}(a, pw), and $\delta 1$ can be multiplied by the cross-section of the logic gate to give the effective cross-section:

$$\sigma_{eff} = \sigma \delta 1. \qquad (4.3)$$

The total effective cross-section of a logic path is the sum of the effective cross-sections of each of the sensitive regions in the logic path. For a logic path with m sensitive regions:

$$\sigma_{eff,\ logic\ path} = \Sigma\sigma_n\ \delta 1_n,\ n= 1 \text{ to } m. \qquad (4.4)$$

Figure 25 shows the logic path from the output of registers A and B in the register file to the input of the ALU register for the AND instruction. The importance of modeling the logic propagation of the Mux2 is apparent in this figure. The sensitive regions are determined by the datapath steered by the Mux2s, which are controlled by the instruction being executed. Table 10 shows $\delta 1$ and the effective cross-section evaluated at each logic block in the path. Appendix B shows this analysis for other logic paths in the KDLX.

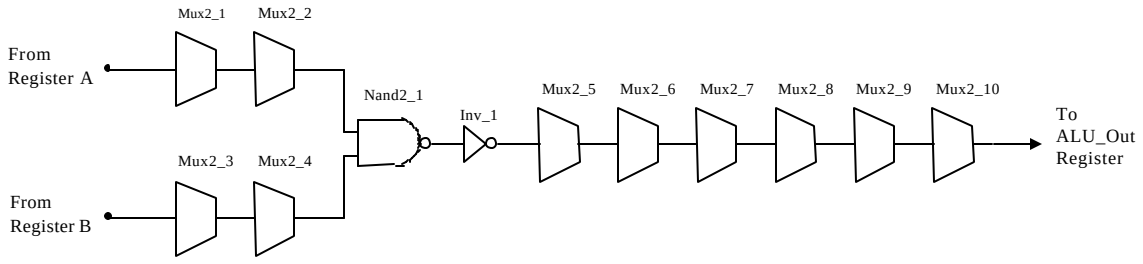


Figure 25. AND Combinational-Logic Datapath

Logic Block	Cross-section (μm^2)	P_{scl}	P_{ap}	$P_{latched}$ (1/MHz)	$\delta 1$ (1/MHz)	Effective Cross-Section $s * \delta 1$ (μm^2/Mhz)	Effective Cross-Section @ 625 KHz	Effective Cross-Section @ 5 MHz
Mux2_1	131.62	0.5	1	1.50E-04	7.50E-05	9.87E-03	3.29E-03	2.63E-02
Mux2_2	131.62	0.5	1	1.50E-04	7.50E-05	9.87E-03	3.29E-03	2.63E-02
Mux2_3	131.62	0.5	1	1.50E-04	7.50E-05	9.87E-03	3.29E-03	2.63E-02
Mux2_4	131.62	0.5	1	1.50E-04	7.50E-05	9.87E-03	3.29E-03	2.63E-02
Nand2_1	41.75	1	1	1.50E-04	1.50E-04	6.26E-03	2.09E-03	1.67E-02
Inv_1	35.79	1	1	1.50E-04	1.50E-04	5.37E-03	1.79E-03	1.43E-02
Mux2_6	131.62	1	1	1.50E-04	1.50E-04	1.97E-02	6.58E-03	5.26E-02
Mux2_7	131.62	1	1	1.50E-04	1.50E-04	1.97E-02	6.58E-03	5.26E-02
Mux2_8	131.62	1	1	1.50E-04	1.50E-04	1.97E-02	6.58E-03	5.26E-02
Mux2_9	131.62	1	1	1.50E-04	1.50E-04	1.97E-02	6.58E-03	5.26E-02
Mux2_10	131.62	1	1	1.50E-04	1.50E-04	1.97E-02	6.58E-03	5.26E-02
Total						**1.50E-01**	**4.99E-02**	**4.00E-01**

Table 10. Effective Cross-Section of AND Datapath

5. Determination of the $\varepsilon 2$ Transitional Probability

The $\varepsilon 2$ transition was determined in a similar manner as the analog propagation simulations. An SET pulse was used to drive the input of the output buffer, which was connected to an 8 pF capacitor (similar to the $\beta 3$ simulations). The resulting pulsewidth and amplitude are shown in Table 11.

Table 11 shows that it takes a very long SET pulse (1430 picoseconds) to propagate to the output with an amplitude of –1.37 volts (with respect to Vdd, or 3.3 Volts). The maximum input voltage that the Xilinx Virtex FPGA will read as a logic "0" is 0.8 Volts [42]. The resulting output of the KDLX is 1.93 Volts (3.3 – 1.37), thus the transient will not be read as logic "0" by the FPGA. Similarly, the minimum voltage that the Xilinx Virtex FPGA will read as a logic "1" is 2.0 Volts, and the resulting amplitude due to a 1370 picosecond length SET is 0.81 volts, so the FPGA will not read the SET as a logic "1." These two cases indicate that even very long SETs will not cause the SET to be read incorrectly by the external system. Thus, $\varepsilon 2$ is set to "0."

Input Amplitude (Volts)	Input Pulsewidth (picoseconds)	Output Amplitude (Volts)	Output Pulsewidth (picoseconds)
-3.3	780	-0.46	1850
-3.3	1010	-0.78	2020
-3.3	1120	-0.93	2150
-3.3	1230	-1.08	2290
-3.3	1320	-1.22	2320
-3.3	1430	-1.37	2380
3.3	750	0.41	1130
3.3	960	0.55	1250
3.3	1080	0.62	1320
3.3	1180	0.69	1430
3.3	1290	0.76	1440
3.3	1370	0.81	1520

Table 11. SET Propagation – Output Buffer

D. SEU PROPAGATION MODELING

1. Objective

The objective of the SEU propagation modeling is to determine the probability that an SEU will propagate to cause an output error. Specifically, it addresses the $\varepsilon 1$ transitional probability. Instruction-based register-usage analysis is used to determine which internal registers are sensitive to an SEU during the execution of an instruction. In most internal registers, it is obvious when an SEU will prevent the proper execution of an instruction. This is not the case for the pipeline registers. Depending on the instruction decoding, an SEU in a pipeline register may or may not prevent proper execution of an instruction. Because of this, VHDL fault-injection modeling is used to determine the effect of an SEU in the pipeline registers.

2. Instruction-Based Register-Usage Analysis

The purpose of instruction-based register-usage analysis is to determine which internal registers are necessary for the proper execution of an instruction. Proper

execution is defined as follows: for each pipeline stage, if all internal registers and external signals that are affected by the instruction are correct at the end of that stage, then proper execution of that stage has occurred. For example, in the register add instruction (ADD Rd, Rs1, Rs2), the contents of source register 1, Rs1, is added to source register 2, Rs2, and stored in the destination register, Rd. Table 12 shows the critical registers for each pipeline stage.

Pipeline Stage	**Critical Registers**
Fetch	Program_Counter
Decode	Decode_Instr_Reg Rs1 Rs2
Execute	Execute_Instr_Reg RA RB
Memory	Memory_Instr_Reg ALU_Out
Writeback	WB_Instr_Reg Delayed_ALU_Out

Table 12. Critical Registers for ADD Rd, Rs1, Rs2 Instruction

If an error occurs in one of the registers during a particular pipeline stage, improper instruction execution will occur. This approach is complete for all registers that are updated every clock cycle, but not all registers are updated every clock cycle. The Rs1 and Rs2 registers (which correspond to registers R1 to R15 in the KDLX) are examples of this. If an error has occurred in either of these registers since they were last written to, improper execution will occur: the result in Rd will not be correct. Thus, it is also necessary to determine the number of clock cycles that have occurred since the register was last updated. Table 13 shows a revised version of Table 12 that includes the number of sensitive clock cycles given register Rs1 was last updated *n* clock cycles ago, and Rs2 was last updated *m* clock cycles ago.

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter: 1 clock cycle
Decode	Decode_Instr_Reg: 1 clock cycle Rs1: *n* clock cycles since Rs1 was last written Rs2: *m* clock cycles since Rs1 was last written
Execute	Execute_Instr_Reg: 1 clock cycle RA: 1 clock cycle RB: 1 clock cycle
Memory	Memory_Instr_Reg: 1 clock cycle ALU_Out: 1 clock cycle
Writeback	WB_Instr_Reg: 1 clock cycle Delayed_ALU_Out: 1 clock cycle

Table 13. Critical Registers and Clock Cycles for Add Rd, Rs1, Rs2 Instruction

3. VHDL Fault-Injection Modeling

The pipeline module of the KDLX processor decodes the 8-bit opcode and provides control signals to the rest of the processor during the decode, execute, memory, and writeback pipeline stages. Depending on the decoding, an SEU may cause an error in these control signals. The purpose of the fault-injection modeling is to determine exactly which bits of the opcode in each pipeline stage can tolerate an SEU without causing an error in the control signals. These results will then be used to further refine Table 13.

The VHDL fault-injection circuit is shown in Figure 26. The blocks labeled "Pipeline_A" and "Pipeline_B" are gate-level VHDL descriptions of the KDLX pipeline module. The block labeled "Opcode_Error_Inject" outputs the byte-wide exclusive-or of the signal Opcode_A_In (the input to Pipeline_A) and the signal Mask_In. This allows errors to be injected into the opcode by changing the bits of Mask_In. The block labeled "Pipeline_Error_Check" compares the outputs of Pipeline_A to the outputs of Pipeline_B. If the injected bit error does not cause a miscompare, all outputs of Pipeline Error Check will be a logic '0'. If the injected error does cause a miscompare, the outputs of Pipeline_Error_Check show which pipeline signal the error occurs in; it also shows what type of error is caused: a PC_Error, a Control_Error, or an Access_Error.

AVHDL test bench works as follows: for every KDLX opcode, the Mask_In signal cycles through the following sequence:

1. “00000001”
2. “00000010”
3. “00000100”
4. “00001000”
5. “00010000”
6. “00100000”
7. “01000000”
8. “10000000”.

Thus, for every opcode, the simulation is run with a single-bit error occurring at each bit location of the opcode. A single-bit error is assumed, because the transistors of the flip-flops are separated enough spatially such that a single particle will not cause a multiple-bit error. By summing the number of bit errors that cause an output error, the cross-section of the opcode register during a particular pipeline stage can be determined. Table 14 shows the results for seven opcodes of this simulation. For each pipeline stage, the number of bit errors that resulted in an output error is shown. The “P” in each column refers to a program address error. The “C” refers to a control error. The “A” refers to an access error.

		Decode Stage			**Execute Stage**			**Memory Stage**			**Writeback Stage**		
Instruction	Opcode	P	C	A	P	C	A	P	C	A	P	C	A
SW	0x45	0	0	8	0	0	7	0	8	0	0	0	8
LW	0x44	0	0	1	0	0	7	0	8	0	0	0	8
J	0xC8	0	0	0	6	0	8	0	0	0	0	0	7
JAL	0xE8	0	0	0	6	0	8	0	0	0	0	0	8
BEQZ	0xC1	0	0	0	7	0	8	0	0	0	0	0	6
BNEZ	0xC0	0	0	0	7	0	8	0	0	0	0	0	5
ADD	0x01	0	0	0	0	0	7	0	0	0	0	0	2

Table 14. Sensitive Bits in Pipeline Registers

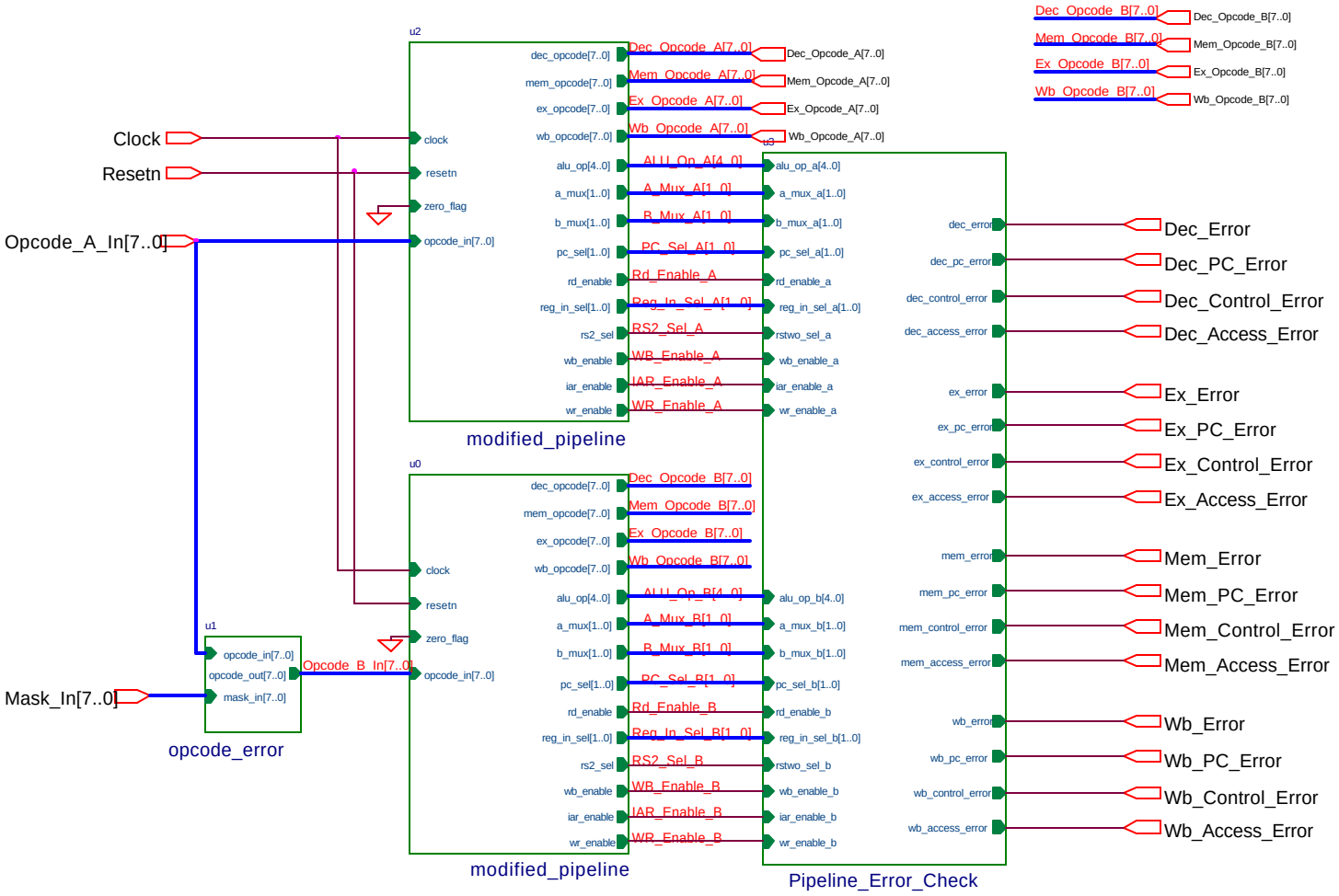


Figure 26. VHDL Fault-Injection Circuit

Table 15 merges the results of the fault-injection modeling with Table 13. This gives the exact number of bits that are sensitive to an SEU during the execution of this instruction.

Pipeline Stage	**Critical Registers & Clock Cycles**
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg12 bits): 1 clock cycle Rs1(16 bits): *n* clock cycles since Rs1 was last written Rs2(16 bits): *m* clock cycles since Rs2 was last written
Execute	Execute_Instr_Reg(11 bits): 1 clock cycle RA(16 bits): 1 clock cycle RB(16 bits): 1 clock cycle
Memory	Memory_Instr_Reg(4 bits): 1 clock cycle ALU_Out(16 bits): 1 clock cycle
Writeback	WB_Instr_Reg(6 bits): 1 clock cycle Delayed_ALU_Out(16 bits):1 clock cycle

Table 15. Critical Bits and Clock Cycles for ADD Rd, Rs1, Rs2 Instruction

4. Determination of the Transitional Probability $\varepsilon 1$

As previously discussed, the transitional probability $\varepsilon 1$ is the probability that an SEU will propagate to the output of the component under test and cause an error in the external system. It is determined by using the results of the preceding two sections applied to the programs of interest. A simple example will illustrate how this is accomplished. Figure 27 shows a simple KDLX program. Using this program, the transitional probability that an SEU in register R1 will propagate to the output ($\varepsilon 1_{R1}$) is determined. Figure 28 shows the sensitive window for register R1. It is labeled t_{sw}. The transitional probability $\varepsilon 1$ is the product of two probabilities:

1. The probability that an SEU occurs in register R1 during the time $t_{sw.}$.
2. The probability that R3 is written to memory.

The probability that an SEU occurs in register R1 during the time t_{sw} is equal to the ratio of t_{sw}/time period of interest. Since the SW instruction comes after the ADD instruction, the probability in 2 is set to 1. Then,

$$\varepsilon 1_{R1} = (t_{sw}/\text{time of interest}) * 1. \quad (4.5)$$

```
LW R0(0), R1; (Loads R1 with Memory[0])
LW R0(1), R2; (Load R2 with Memory[1])
NOP
NOP
NOP
NOP
ADD R3, R1, R2; (Adds R1+R2, writes the sum to R3)
NOP
NOP
NOP
NOP
SW R0(2), R3;    (Stores contents of R3 to Memory[2])
```

Figure 27. Example Program

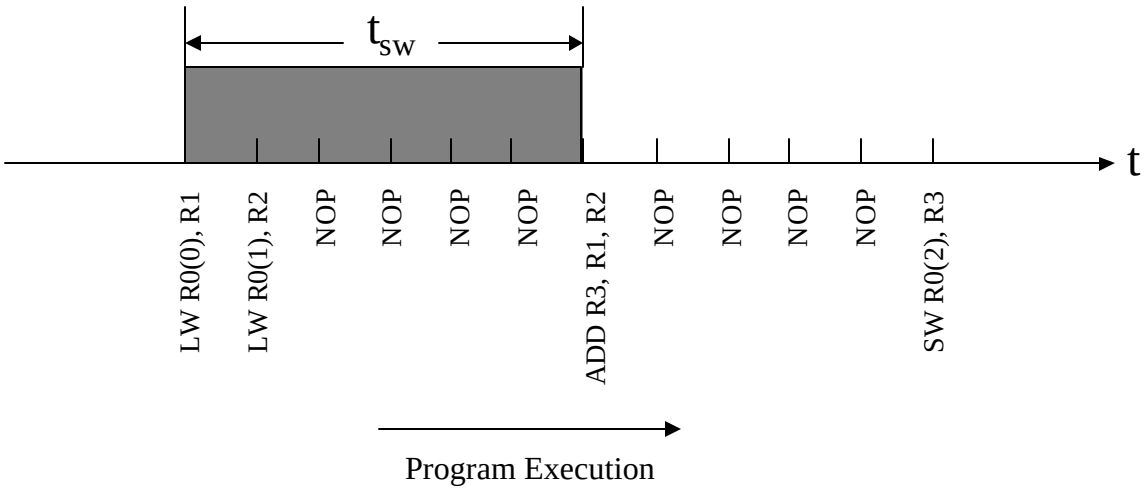


Figure 28. Sensitive Window for R1

E. SYSTEM-LEVEL PREDICTION

The results of the SET propagation simulations and SEU propagation modeling are applied to determine the effective cross-sections of three test programs. The effective cross-section for a given program is the product of the SET transitional probabilities, the SEU transitional probabilities and the cross-sections determined in the SET generation modeling. Test Program #1 is a program that loads all the registers, writes them out immediately, waits for approximately 240 clock cycles, and repeats the

process. Test Program #2 is similar, except it loads all registers, waits for 240 clock cycles, and writes them out. Test Program #3 is a functionality test program, similar to the program used for verification of the processor design prior to fabrication. Test Program #3 loads the registers, performs an operation (e.g., ADD, XOR) on the register, and writes the result to the output. All operations are exercised in this manner in Test Program #3. Table 16 shows the average number of sensitive bits per clock cycle for each program. Table 17 shows the contribution of the memory elements and logic elements to the effective saturated access error cross-sections for Test Program #1 and Test Program #2. The crossover point in the table is the frequency at which the contribution due to logic elements is equal to the contribution due to memory elements. The table shows that the effective cross-section due to the logic elements is negligible at 625 kHz and 5 MHz. Figure 29 shows the predicted access-error cross-section as a function of LET. Figure 30 shows the predicted control-error cross-section as a function of LET, and Figure 31 shows the predicted program-address-error cross-section as a function of LET.

Test Program	**Access-Errors Sensitive Bits per Clock Cycle:**	**Control-Errors Sensitive Bits per Clock Cycle:**	**Program-Address-Errors Sensitive Bits per Clock Cycle:**
1	10.7	1.2	15.94
2	231.0	1.2	15.94
3	272.0	7.13	19.24

Table 16. Average Number of Sensitive Bits per Clock Cycle

Test Program	Effective Cross-section Due to Memory Elements (cm^2/device)	Effective Cross-section Due to Logic @ 625 kHz (cm^2/device)	Effective Cross-section Due to Logic @ 5 Mhz (cm^2/device)	Cross-over Frequency
1	3.59e-6	9.375e-10	7.5e-9	2.393 GHz
2	7.77e-5	9.375e-10	7.5e-9	51.8 GHz

Table 17. Comparison of Memory-Element Versus Logic-Element Saturated Access Error Cross-Sections

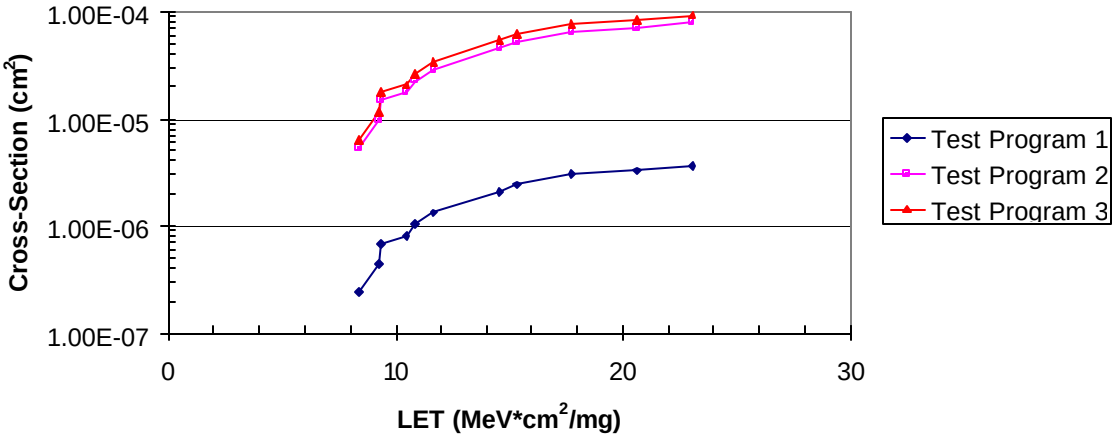


Figure 29. Predicted Access-Error Cross-Section Versus LET

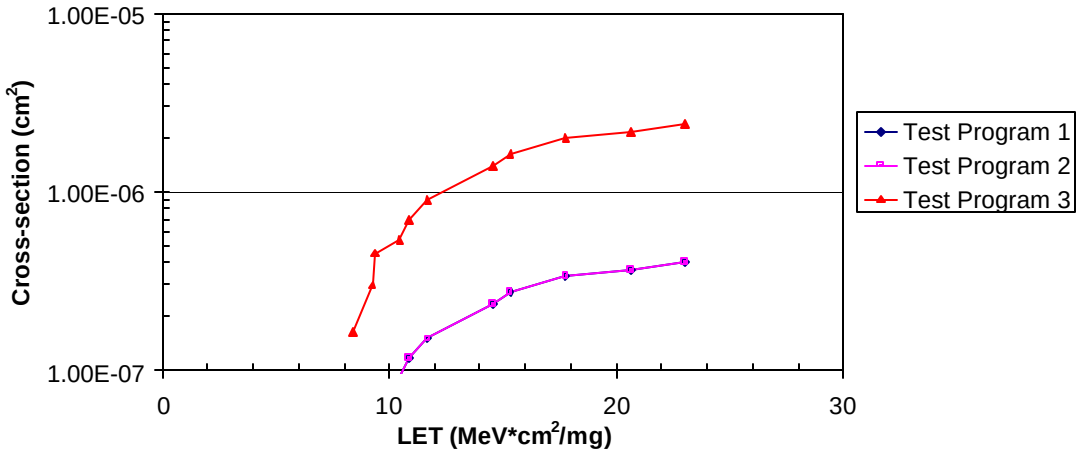


Figure 30. Predicted Control-Error Cross-Section Versus LET

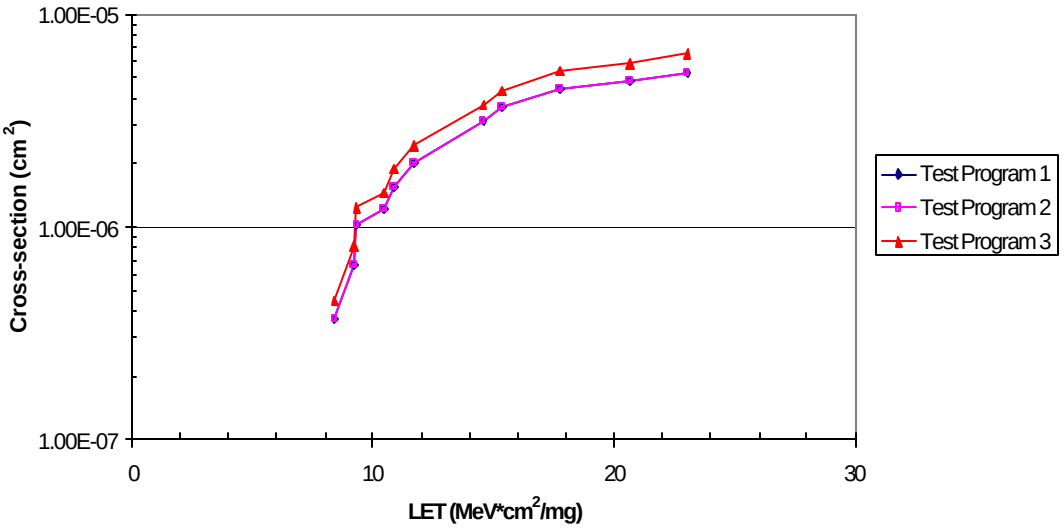


Figure 31. Predicted Program-Address-Error Cross-Section Versus LET

THIS PAGE INTENTIONALLY LEFT BLANK

V. MODELING VALIDATION

A. OBJECTIVE

The objective of this chapter is to document the validation of the modeling results from Chapter IV. This was accomplished by performing laser and heavy-ion testing on the KDLX processor. The laser provides the means to inject an SET directly on a particular transistor. This allows for direct validation of the key transitional probabilities $\delta 1$ and $\varepsilon 1$. Heavy-ion testing provides SET injection that is both spatially and temporally random. It is not known exactly which transistor will have an SET. This provides the means to validate the system-level predictions. Additionally, it provides the opportunity to evaluate some of the predicted physical parameters: saturated cross-section and onset LET. This chapter describes the test system, the laser testing results, and the heavy-ion testing results.

B. TEST SYSTEM

1. Objective

The objective of the test system is to provide a means of capturing all required information during a test. This information must describe an error that has occurred in the KDLX processor during laser and heavy-ion testing. The address bus, the data bus, the program address bus, and the read and write control signals are the required information.

2. Description

Figure 32 shows the configuration of the test system. It consists of the laser or heavy-ion beam source, a personal computer (PC), and the test board. The details of the laser and heavy-ion beam source will be discussed in later sections. The PC controls the operation of the test board and records the test results. The test board is described below.

Figure 33 shows a block diagram of the test board. Conceptually, it consists of two pieces, the KDLX device under test and the KDLX_Tester FPGA. The KDLX_Tester FPGA provides test control and implements the “golden chip” method of processor testing described in Koga [14]. The FPGA contains a functionally-equivalent VHDL description of the KDLX, the Golden Chip KDLX. The program and data

memories are provided in the FPGA. The Comparison Logic Module captures the program address bus, the address bus, the data bus, the read signal and the write signal from both the KDLX under test and the Golden Chip KDLX on every clock cycle. These values are compared. If there is a difference between the KDLX under test and the Golden Chip KDLX, an error flag goes high and the captured values are written to the Error FIFO. The Error FIFO provides temporary storage for the error data. The Error Counters Module contains three counters: the access error counter, the program address error counter, and the control error counter. These counters are incremented depending on the type of error. The Universal Asynchronous Receiver/Transmitter (UART) provides the interface to the PC.

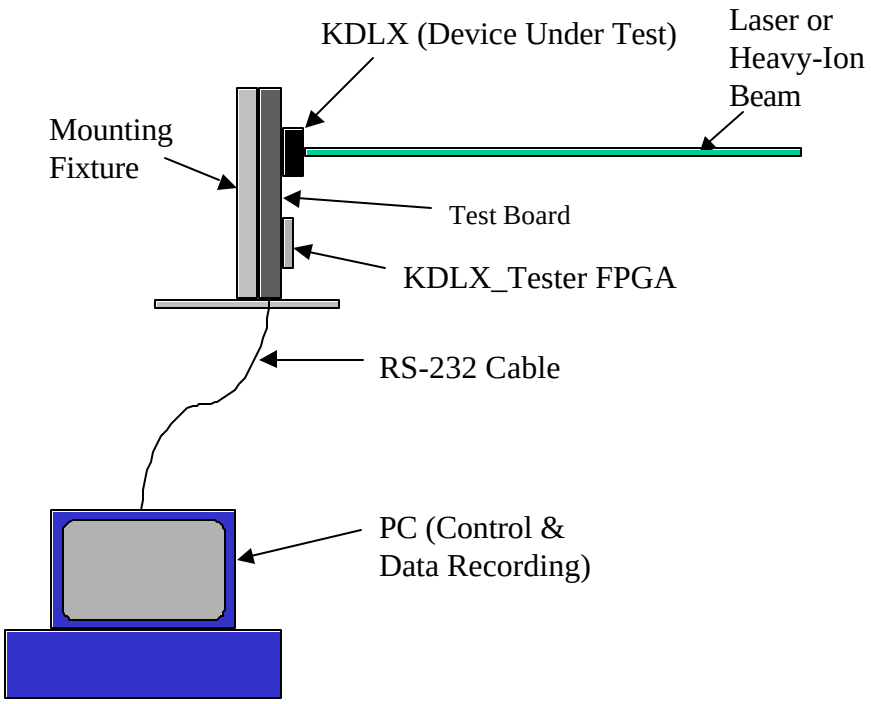


Figure 32. Test Configuration

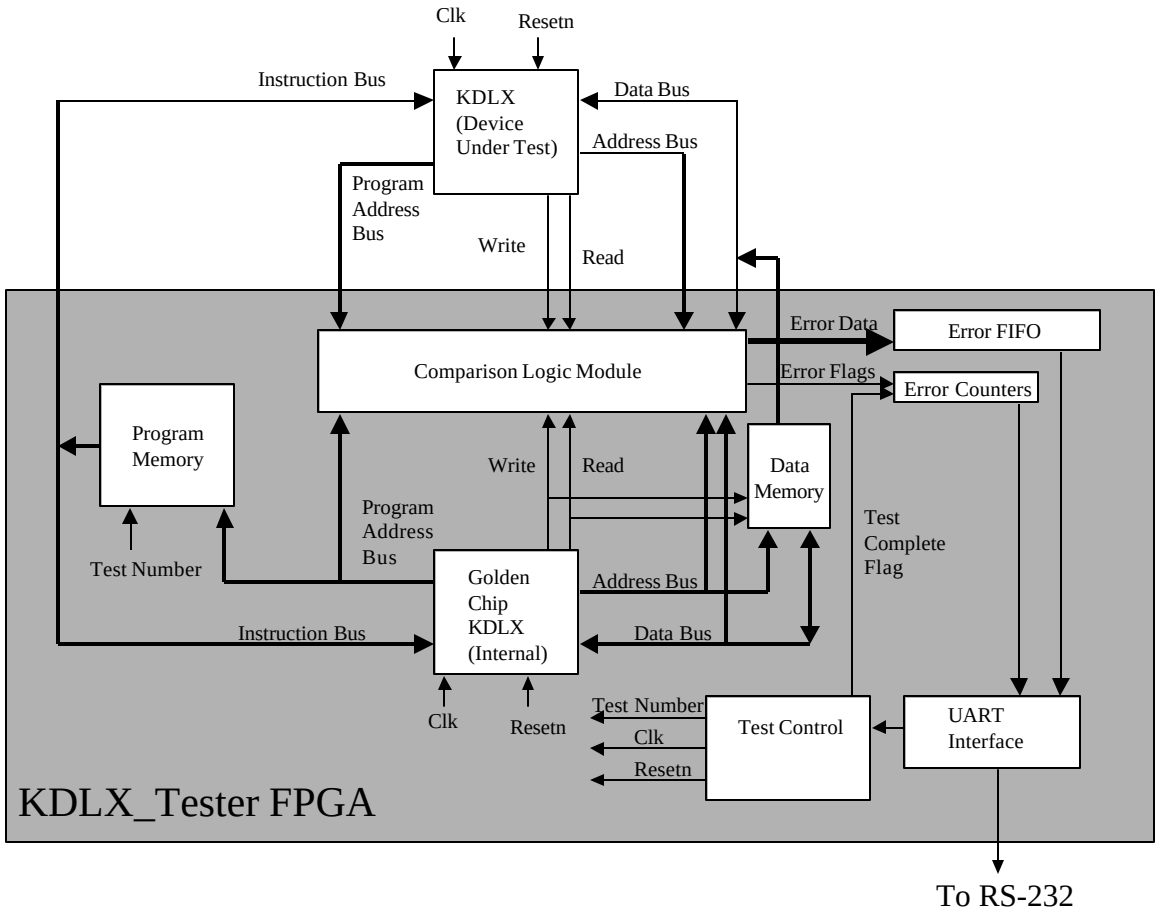


Figure 33. Test-Board Block Diagram

3. Operation

The test board is placed in a position such that the laser or heavy-ion beam will be incident upon the KDLX under-test. The beam is turned on. The PC sends the test command. This specifies the test number, clock speed, and test length. Once this command has been completely received, the Test Control block pulls the Resetn signal to a logic level "0," synchronously resetting both the device-under-test and the Golden Chip. The Test Control block also provides the Clk signal to both processors at the commanded clock speed. The Test Number specifies which of the test programs is used during a particular test. The Test Number drives the upper two address bits of the Program Memory to choose the test program.

When the Resetn signal goes to a logic "1," both processors begin executing the first instruction from the chosen test program in Program Memory. On every clock cycle, the program addresses, read signals and write signals are checked. If the program addresses do not match, a Program-Address Error has occurred. The Program-Address Error Counter is incremented and the outputs of both processors are saved in the Error FIFO. Both processors are then reset to resynchronize their program counters. If the read or write signal does not agree, the Control-Error Counter is incremented and the outputs of both processors are saved in the Error FIFO. Because the processors are still synchronized, they are not reset. If either both read signals or both write signals are active (which indicates an access to the data memory without a control error), the address and data buses are checked. If they do not agree, then an access error has occurred. The Access Error Counter is incremented and the outputs are saved. The Error FIFO passes the data to the UART, which sends it to the PC. This process continues until the number of clock cycles specified in the Test Length has occurred. This marks the end of the test. At this point, the Error Counters block sends the values of its counters to the PC via the UART. The test is complete when the PC receives the counter values. Another test can be run immediately by sending another command from the PC.

C. LASER TESTING

1. Objective

The objective of the laser testing is to validate the predicted transitional probabilities $\delta 1$ and $\varepsilon 1$. The laser provides the opportunity to inject an SET on a specific transistor within the KDLX. Focusing the laser on a transistor of a logic gate allows direct insertion into state S2. This provides for validation of the two critical elements of $\delta 1$: clock-edge effects modeling and the probability of logic propagation. Similarly, focusing the laser on a transistor within a flip-flop provides direct insertion into state S3. This provides for validation of the instruction-based register-usage analysis used to predict $\varepsilon 1$.

2. Test Configuration

The laser tests were performed at the Naval Research Laboratory's Pulsed-Laser Facility for Single-Event-Effects Investigation. A block diagram of the test configuration

is shown in Figure 34. The laser source is a 590 nm wavelength pulsed dye laser. The laser pulses are nominally 1 picosecond in length. Optics between the laser source and the device-under-test focus the beam to a spot size of approximately 1.2 to 1.5 μm [44]. This allows the targeting of a single transistor.

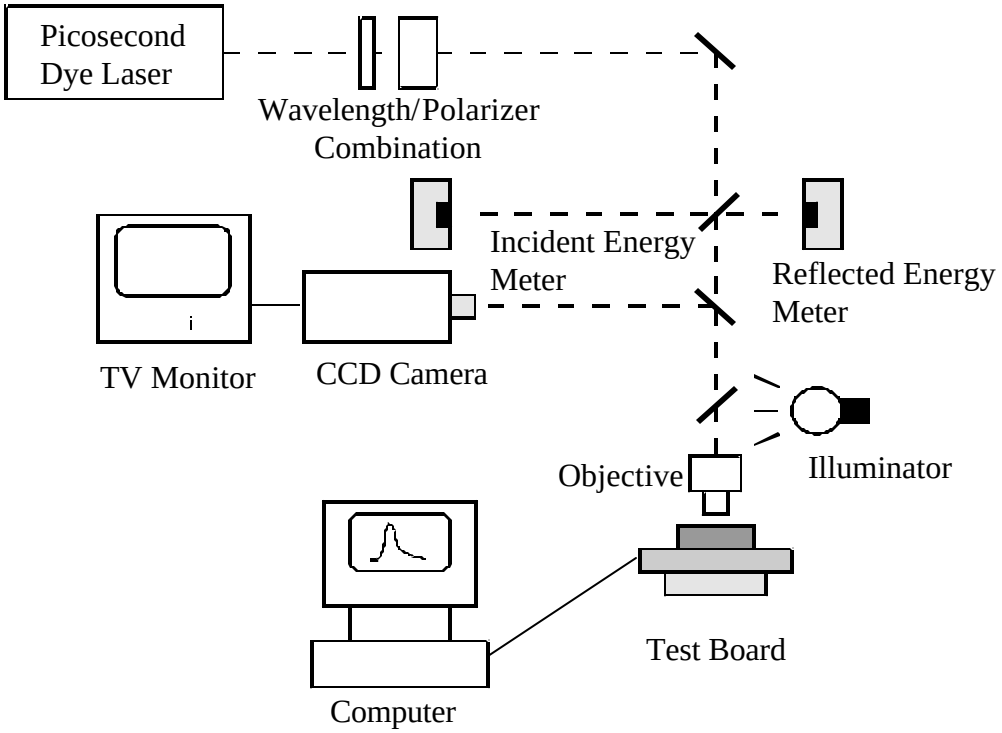


Figure 34. Laser Test Configuration (after [44])

3. Test #1

The purpose of Test #1 is to validate the logic propagation modeling. Specifically, it validates the modeling of logic propagation for the Mux2 standard-cell multiplexer. Table 8 defines this probability as being "Instruction-Dependent." This is a critical concept in the determination of the effective cross-section of a logic path; it says that logic elements that are not in the logic path do not contribute to the effective cross-section. To validate this, the functionality test program, Test Program #3, was executed with the laser beam focused on the combinational-logic elements of the ALU_Logic_Slice module. This module performs the logic operations of the arithmetic logic unit (ALU). Figure 35 shows its schematic. The module consists of an AND gate, an OR gate, an XOR gate, and three multiplexers that determine the output. For logical

AND instructions, Sel0=1 and Sel1=0, steering the output of the AND gate to the output of the module. Similarly, for logical-OR instructions, Sel0=0 and Sel1=1, steering the output of the OR gate to output of the module. The output of the XOR gate is steered to the module output with Sel0=1 and Sel1=1 for exclusive-or instructions.

Figure 36 shows the layout of the ALU_Logic_Slice module. The red circles show the targeted regions (also shown in Figure 35). For Test Run #1, the beam was focused on the output of the AND gate. Ten errors were observed at the output: six occurred during the ANDI instruction execution, and four occurred during the execution of the AND instruction. None occurred during the logical-or (OR, ORI) or the exclusive-or (XOR, XORI) instructions during Test Run #1. In Test Run #2, the beam was focused on the output of the OR gate. Errors occurred only during the execution of the logical-or instructions. Similarly, Test Run #3 focused the beam on the output of the XOR gate. Errors occurred only during the execution of the exclusive-or instructions. The results of these three test runs are summarized in Table 18. These results validate the premise that the combinational-logic elements that are not in an instruction's data path do not contribute to the effective cross-section for that instruction.

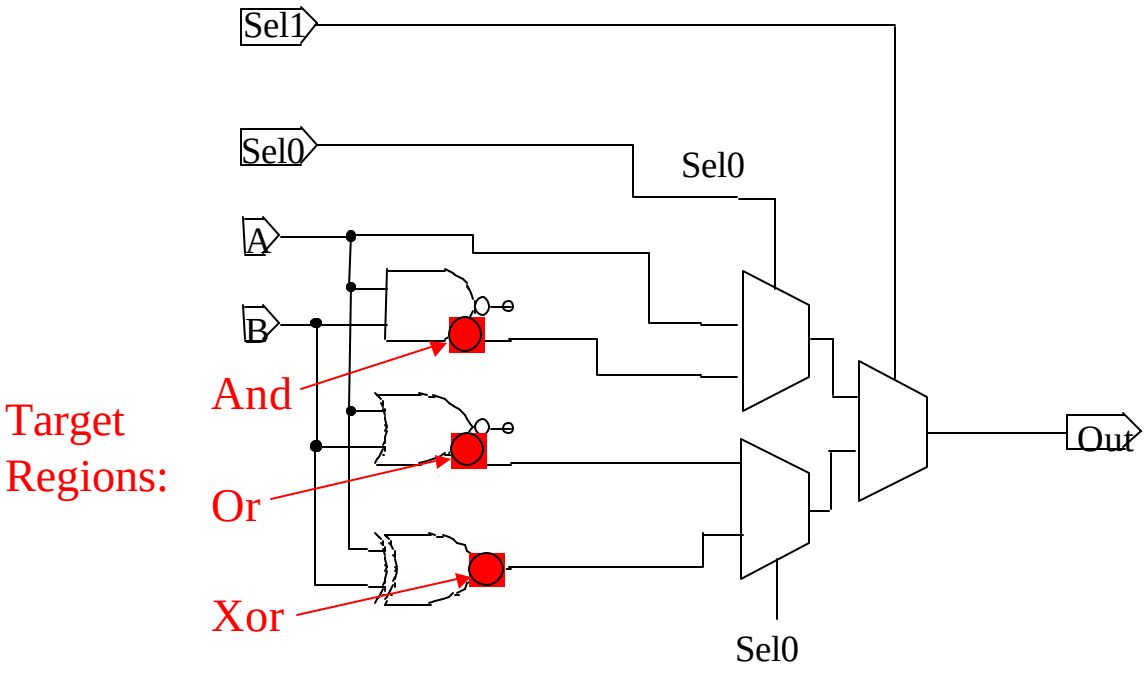


Figure 35. Target Regions for ALU_Logic_Slice

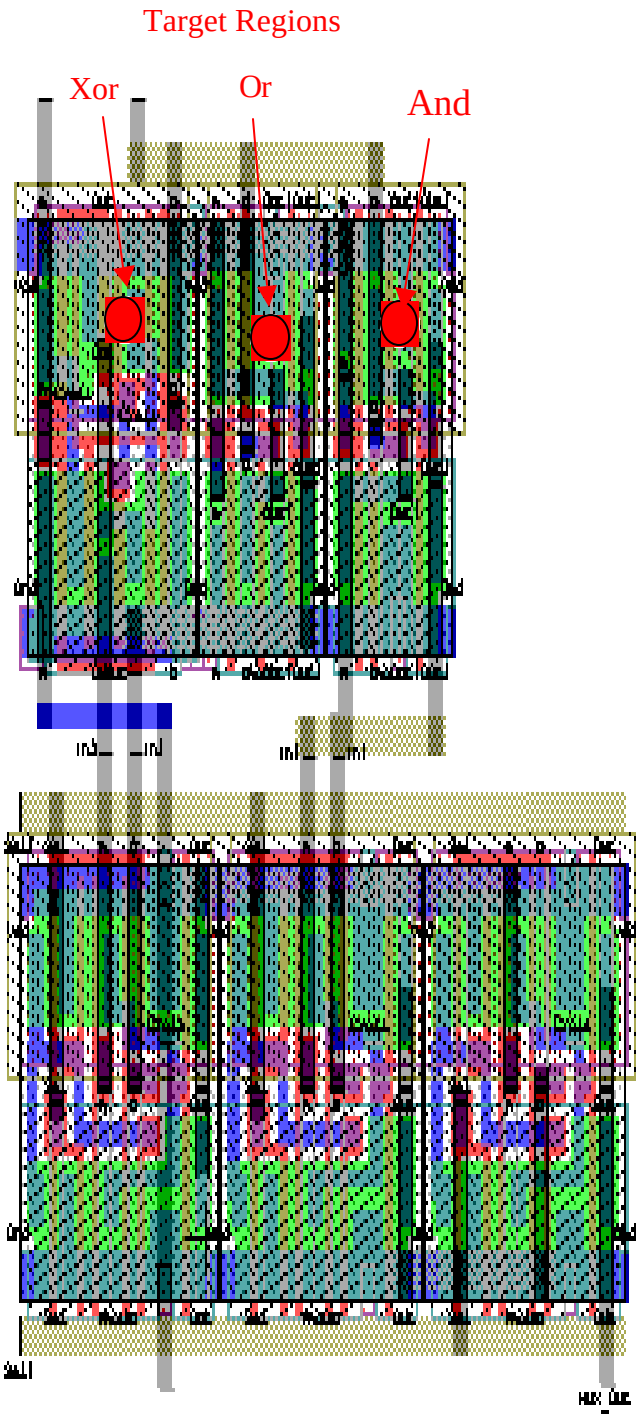


Figure 36. Target Regions for ALU_Logic_Slice – Layout (after [45])

Test Run	Target Cell	Number of Errors	Corresponding Instructions
1	AND Gate	10	ANDI(6) AND(4)
2	OR Gate	11	ORI(5) OR(6)
3	XOR Gate	9	XORI(4) XOR(5)

Table 18. Laser Test #1 Results

4. Test #2

The purpose of Test #2 is to validate the clock-edge effects modeling. Specifically, the relationships among the clock frequency, SET pulsewidth, and the probability that an SET is latched (P_{latch}) are validated. This was accomplished by injecting an SET on a transistor in the second XOR gate of the Full_Adder module, as shown in Figures 37 and 38. In the first group of tests, the output energy detector voltage was 14 mV. In the second group, the laser energy was decreased; the output energy detector voltage was 8 mV. This resulted in a reduced length SET pulse. For each group of tests, the KDLX executed Test Program #2 at four clock frequencies: 625 kHz, 1.25 MHz, 2.5 MHz, and 5 MHz. Table 19 shows the results of these tests. Figure 39 shows a plot of the number of upsets versus the clock frequency. The linear relationship between the clock frequency and the number of upsets is clearly evident, particularly at the higher energy (where the statistics are better). This validates the predicted linear relationship between clock frequency and P_{latch}.

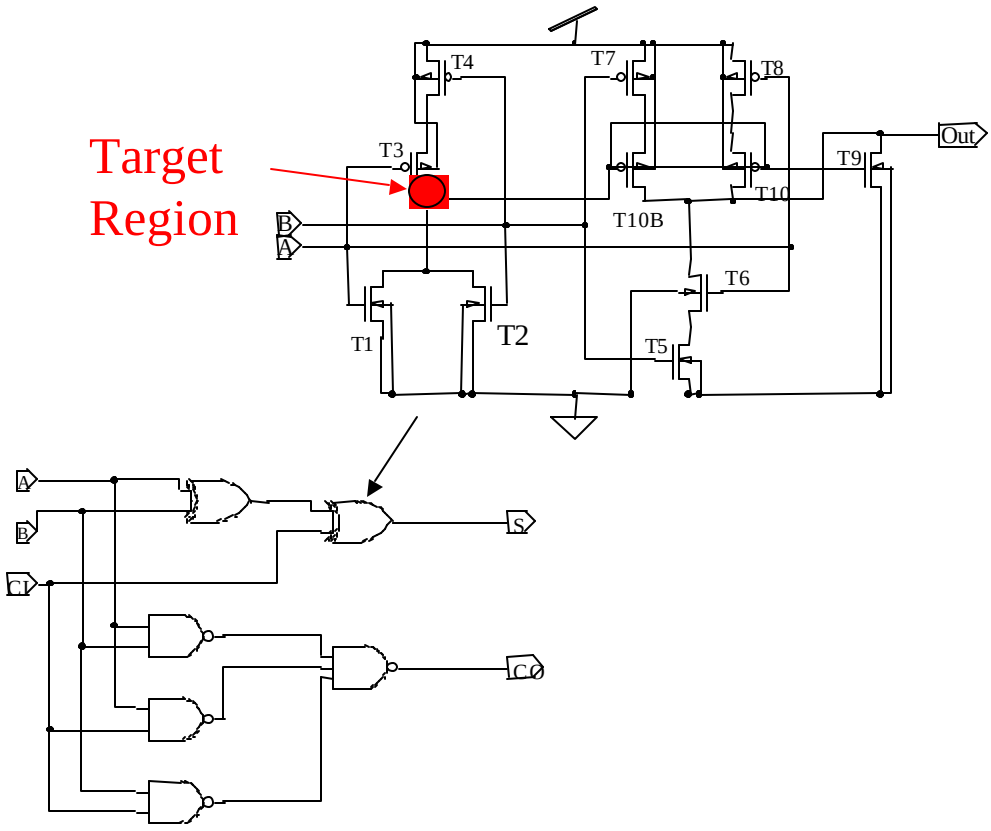


Figure 37. Target Region of Full Adder – Schematic (after [46])

Clock Speed	Laser Energy Detector Output Voltage (mV)	Number of Upsets	P_{latch}
5 MHz	14	138	1.15e-3
2.5 MHz	14	78	6.5e-4
1.25 MHz	14	34.3	2.86e-4
0.625 MHz	14	17.3	1.44e-4
5 MHz	8	16	1.33e-4
2.5 MHz	8	12	1.00e-4
1.25 MHz	8	4	3.33e-5
0.625 MHz	8	2	1.67e-5

Table 19. Laser Test #2 Results

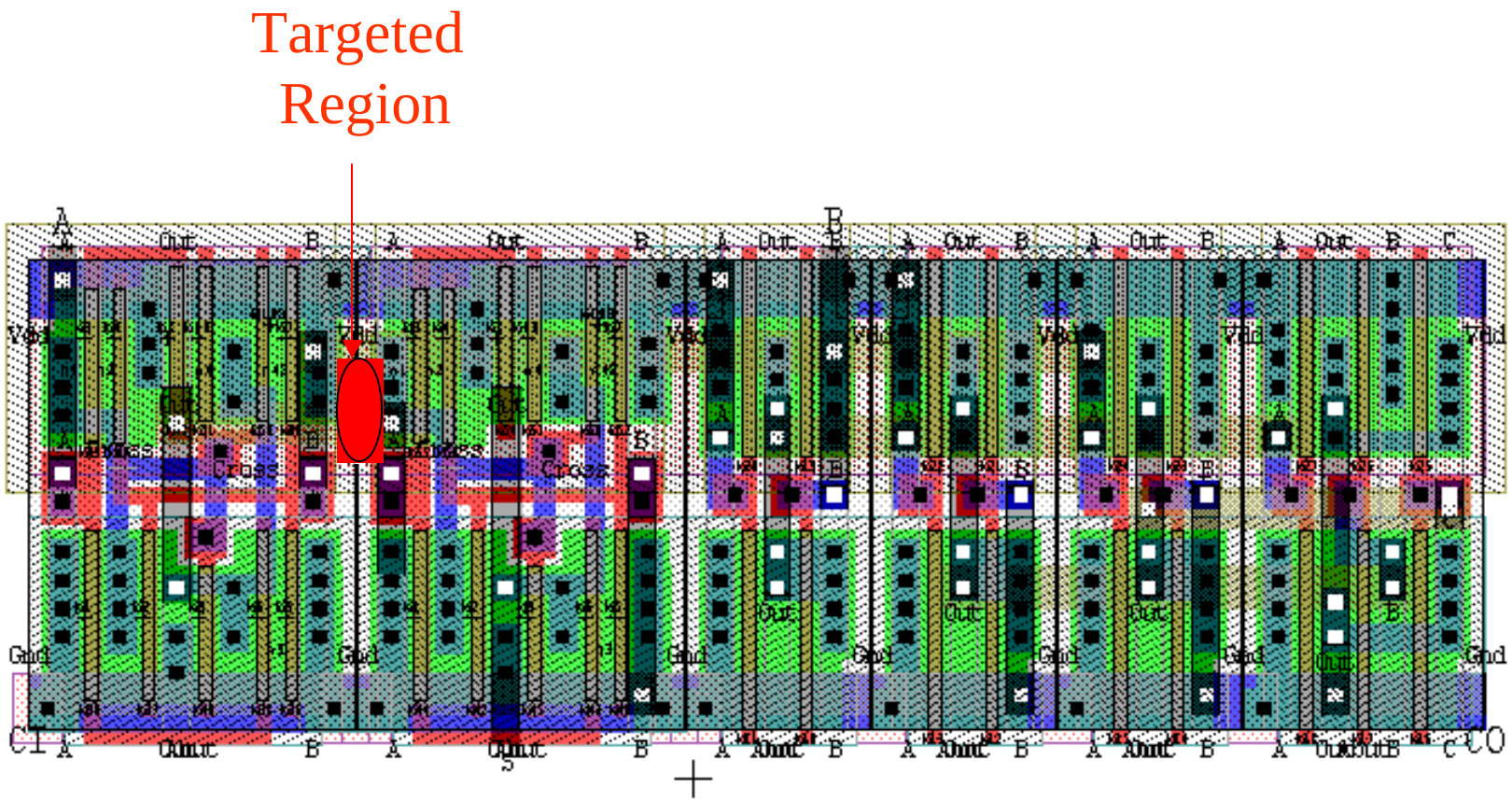


Figure 38. Target Region of Full Adder – Layout (after [45])

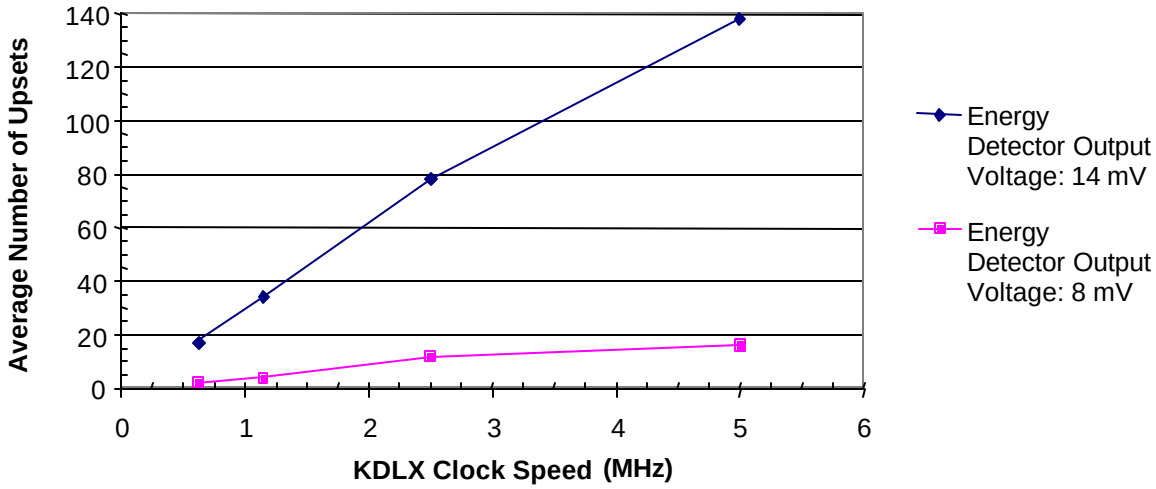


Figure 39. Laser Test #2 Results: Clock-Edge Effects

Validation of the absolute quantitative relationship between the length of the SET pulse and P_{latch} requires an accurate measurement of the SET pulsewidth at the injection node. Unfortunately, this is not possible with the KDLX chip. This is because the SET must propagate through multiple logic gates prior to reaching the output; it is shaped and attenuated during this propagation and thus cannot be accurately measured. However, it is clear from the data that for a given clock rate, a longer SET pulse results in a larger P_{latch}.

5. Test #3

The purpose of Test #3 is to validate the predicted transitional probability $\varepsilon 1$. Validation requires injecting an SEU into a register and observing the resulting number of output errors as a function of the program. To accomplish this, the laser beam was focused on the least-significant bit of register R1 as shown in Figures 40 and 41. This transistor is sensitive only when the clock is high, so the probability of a laser pulse directly causing an SEU is 0.5 (i.e., $\beta 1 = .5$). The pulse repetition frequency of the laser was set at 1 KHz, and Test Program #1 was executed. Each test run lasted two minutes, causing an estimated 60,000 SEUs. This was repeated for Test Program 2, but the laser pulse repetition frequency needed to be reduced to 100 Hz because the test system could not keep up with the error rate. This resulted in an estimated 6000 SEUs. Table 20 shows the test results. The measured transitional probability $\varepsilon 1$ for Test Program 1 was

0.00397. This shows very good agreement with the predicted ε1: 0.00391. For Test Program 2 the measured ε1 was 0.931. This also shows good agreement with the predicted ε1: 0.922. These results validate the modeling approach for the transitional probability ε1.

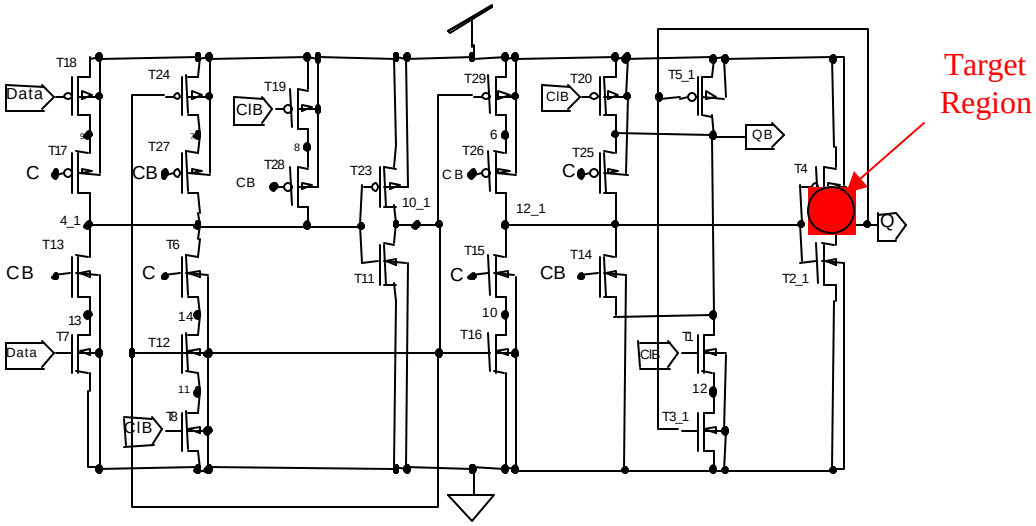


Figure 40. Laser Target Region for DFFC D-Flip-Flop – Schematic (after [41])

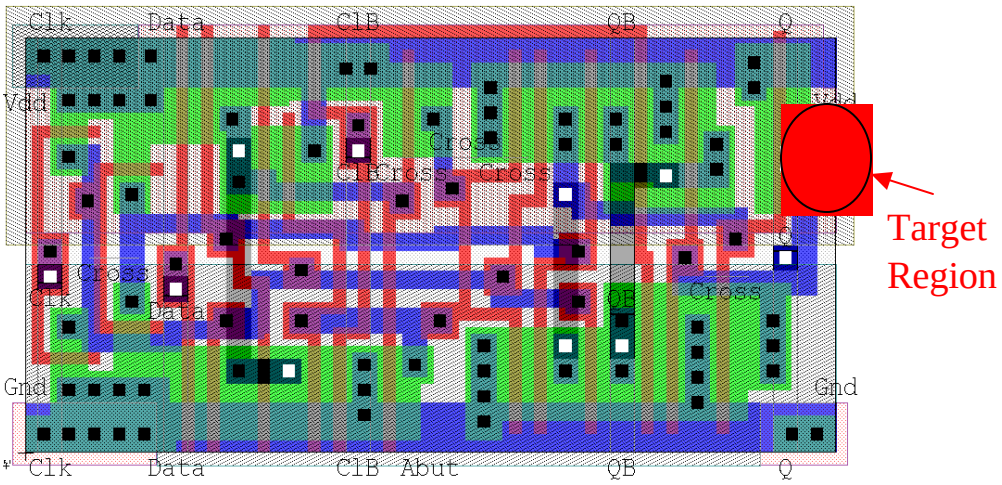


Figure 41. Laser Target Region for DFFC D-Flip-Flop – Layout (after [45])

Program	Laser PRF	Number of Pulses	Estimated Number of SEUs	Average Number of Output Errors	Transitional Probability e1 (Measured)	Transitional Probability e1 (Predicted)
Test Program 1	1 kHz	120,000	60,000	238.4	0.00397	0.00391
Test Program 2	100 Hz	12,000	6000	5479.8	0.9133	0.922

Table 20. Laser Test #3 Results

D. HEAVY-ION TESTING

1. Objective

The objective of the heavy-ion testing is to validate the system-level predictions in Chapter IV. In laser testing, the SET generation is controlled by focusing the beam on the transistor of interest. In heavy-ion testing, the LET and fluence of particles are controlled, but the exact location of ion impact is not. The SET generation is governed by the relationship for equation 3.2:

$$P(\text{SET occurring with given LET}) = \sigma\Phi(LET) \qquad (3.2).$$

Since $\Phi(LET)$ is controlled, the heavy-ion testing provides a measure of the device cross-section as a function of LET. By executing the three different test programs used for the system predictions in Chapter IV, the program-dependent cross-sections can be validated. The predicted saturated cross-section and onset LET can also be validated.

2. Test Operation

The heavy-ion tests were performed at the Texas A & M University Cyclotron Institute Radiation Effects Facility. The same test board and computer used in the laser testing were used. Figure 42 shows a close-up of the output of the beam and the device under test. Three different species of ions were used to provide six different LET values. These are shown in Table 21.

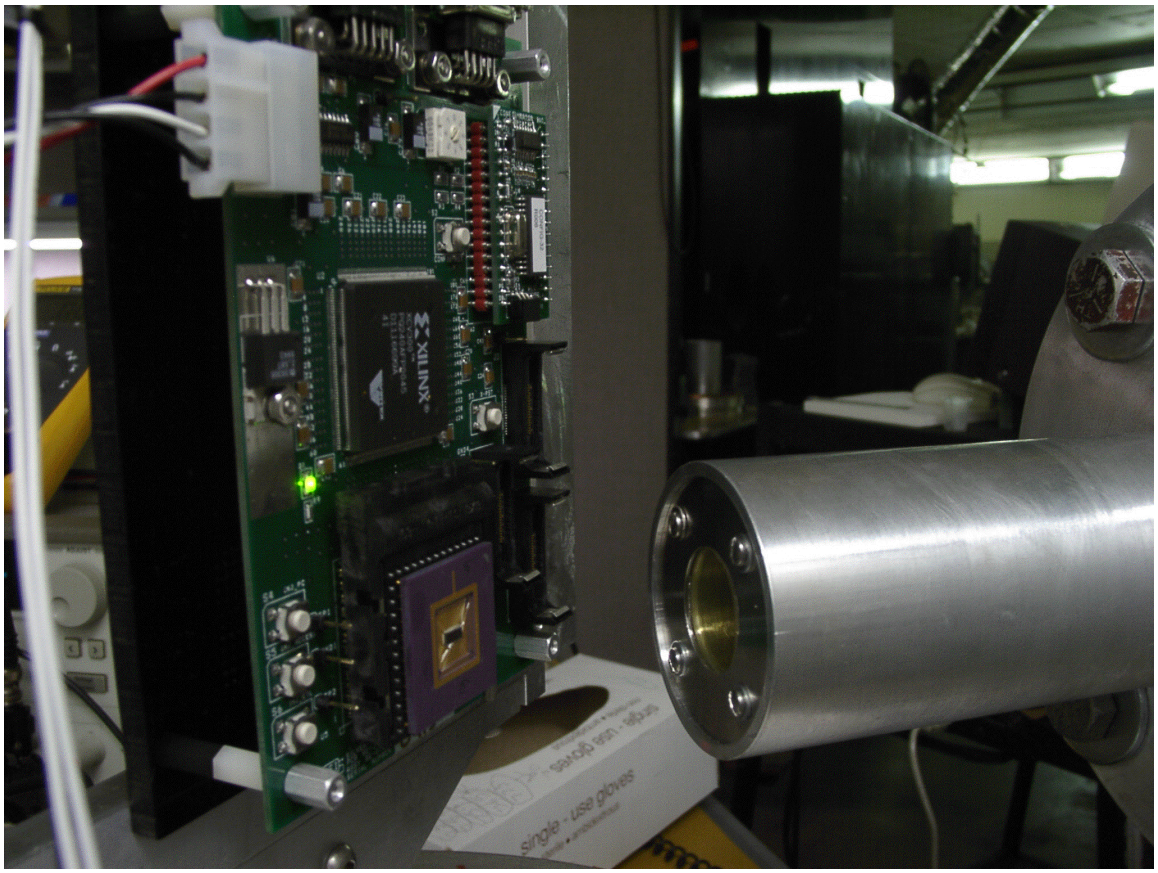


Figure 42. Heavy-Ion Test Configuration

Ion	Energy (MeV)	Angle of Incidence (Degrees)	LET ($MeV*cm^2/mg$)
Argon	933	0	5.69
Argon	933	45	8.08
Krypton	1862	0	20.6
Krypton	1862	45	28.9
Xenon	2730	0	40.9
Xenon	2730	45	57.8

Table 21. Heavy-Ions Used for Heavy-Ion Testing

For each of the LETs shown in Table 21, two test runs were performed: one with KDLX #1, and the other with KDLX #2. Two different KDLX devices were used to

obtain better statistics. Each test run lasted twelve minutes and consisted of the following sequence of test programs:

1. Test Program #1 @ 5 MHz
2. Test Program #2 @ 5 MHz
3. Test Program #3 @ 5 MHz
4. Test Program #1 @ 625 kHz
5. Test Program #2 @ 625 kHz
6. Test Program #3 @ 625 kHz.

Each program lasted exactly two minutes and was run at both the highest and lowest clock frequencies available on the test system. During the execution of each program, the number of access, program address, and control errors was recorded, as was all the error information.

3. Test Results

The test results are divided into three cross-section versus LET plots: access errors, control errors, and program address errors. Figures 43 and 44 show the access-error cross-section versus LET. Figures 45 and 46 show the program-address- error cross-section versus LET. Figures 47 and 48 show the control-error cross-section versus LET.

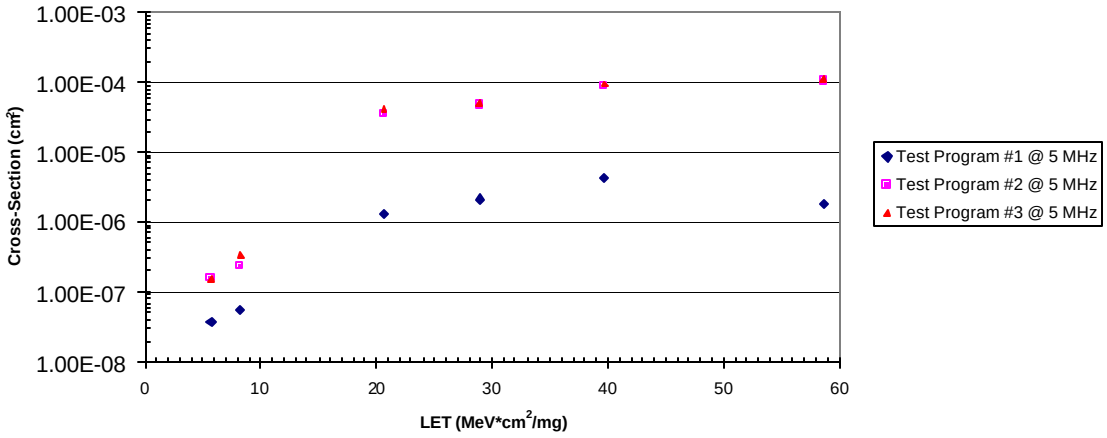


Figure 43. Measured Access-Error Cross-Section Versus LET @ 5 MHz

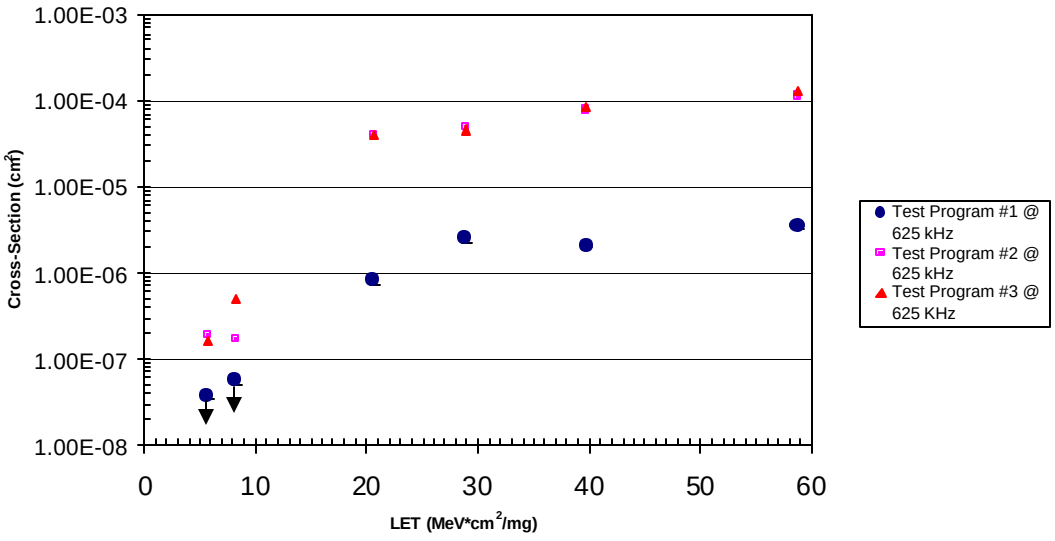


Figure 44. Measured Access-Error Cross-Section Versus LET @ 625 kHz

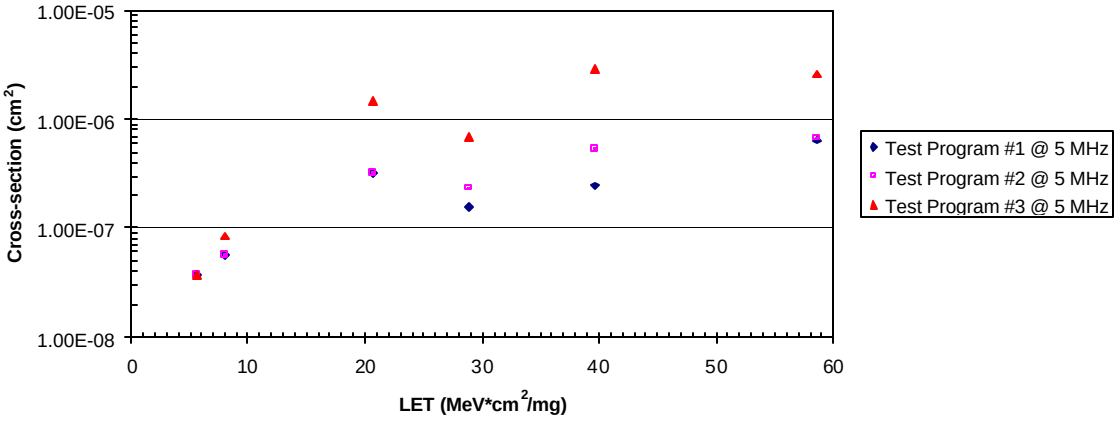


Figure 45. Measured Control-Error Cross-Section Versus LET @ 5 MHz

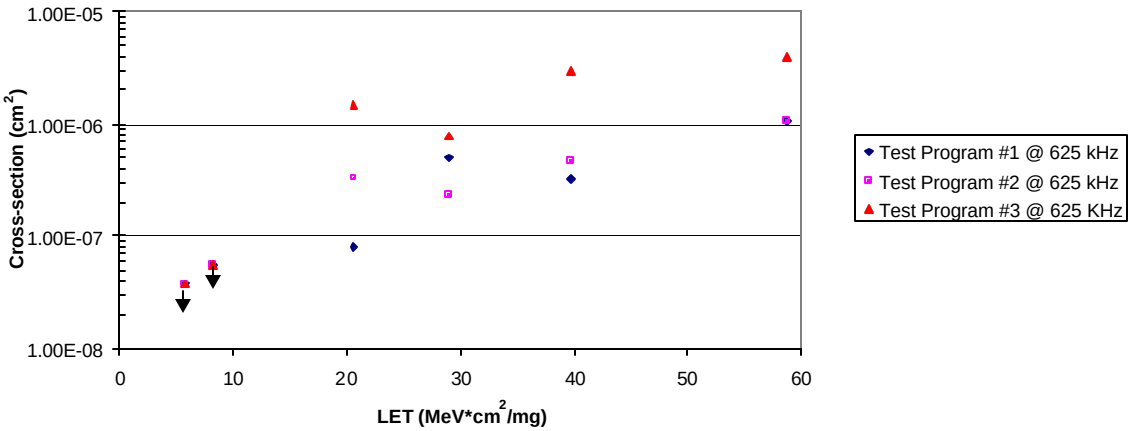


Figure 46. Measured Control-Error Cross-Section Versus LET @ 625 kHz

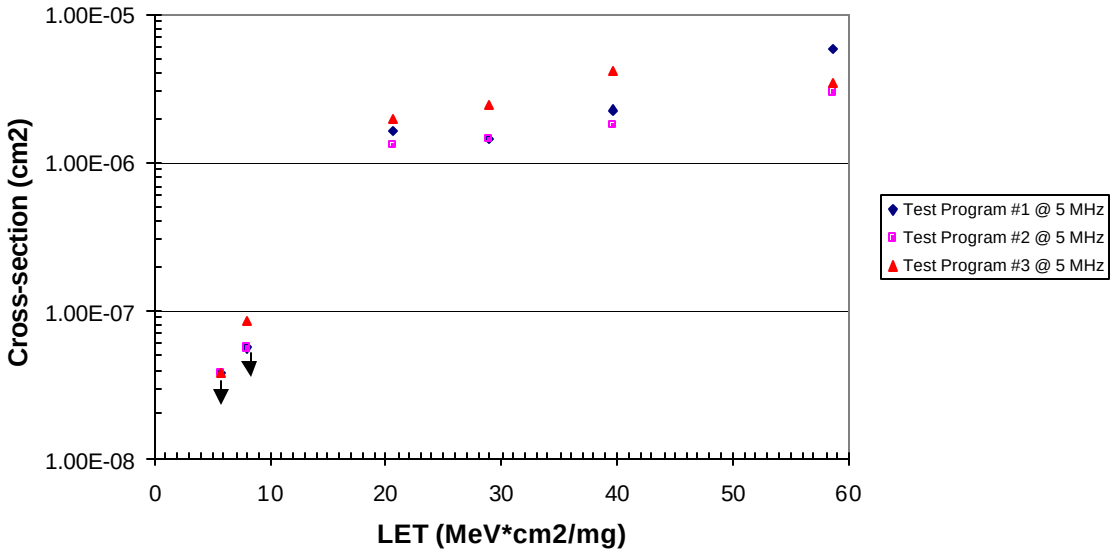


Figure 47. Measured Program-Address-Error Cross-Section Versus LET @ 5 MHz

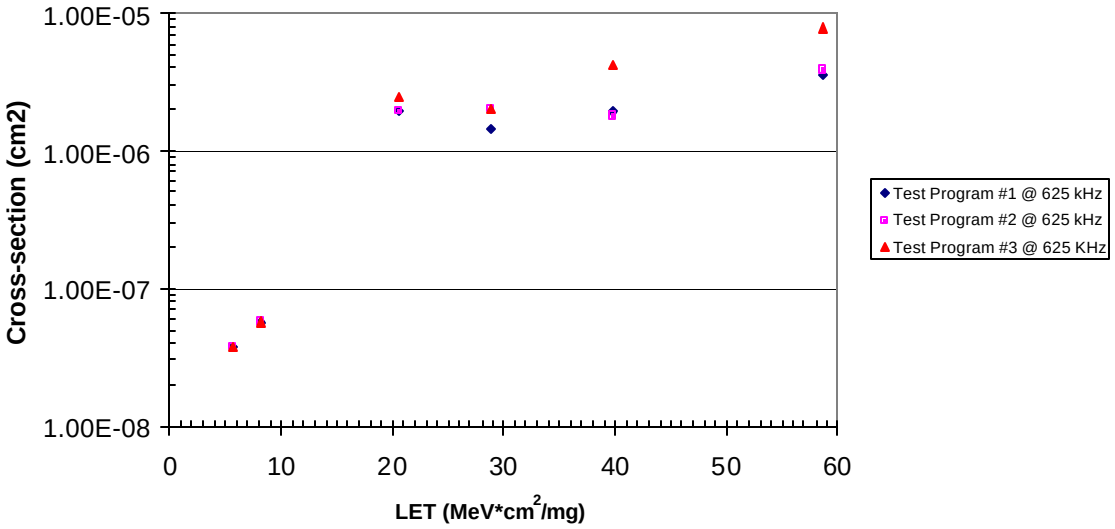


Figure 48. Measured Program-Address-Error Cross-Section Versus LET @ 625 kHz

E. COMPARISON BETWEEN SYSTEM PREDICTIONS AND TEST RESULTS

Figure 49 shows a comparison between the predicted access-error cross-sections and the measured access-error cross-sections from the heavy-ion testing. The Measured Test Program #1 cross-section is the average of the cross-sections of the 625 kHz and 5 MHz tests. This is also true for Test Program #2 and Test Program #3. The predicted cross-sections track the measured values well, especially at the higher LETs.

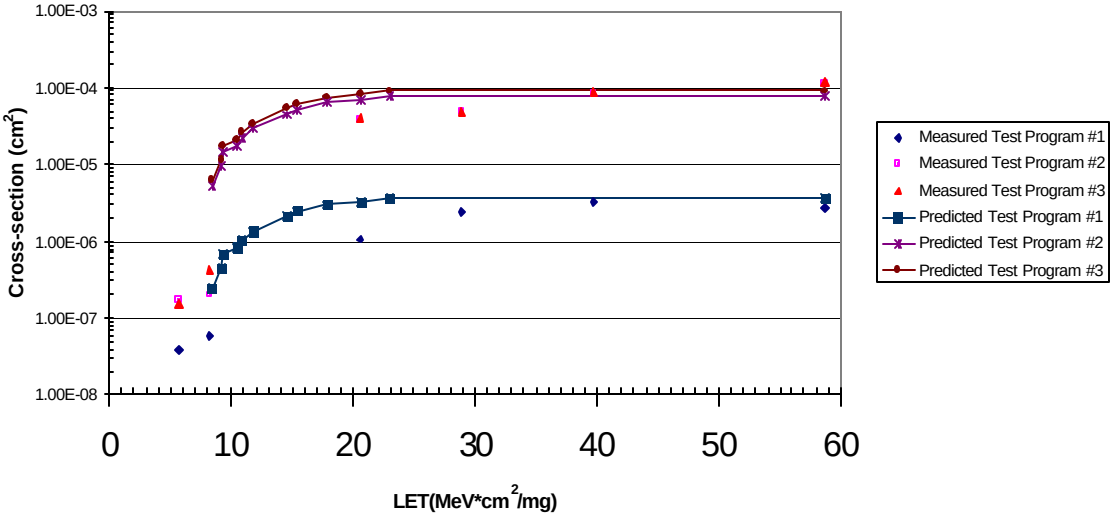


Figure 49. Measured and Predicted Access-Error Cross-Section Versus LET

Table 22 compares the measured and the predicted saturated cross-section and onset LETs for each of the test programs. The predicted saturated cross-sections match the measured cross-sections very well. For each of the test programs, the predicted values are within one standard deviation of the measured cross-section.

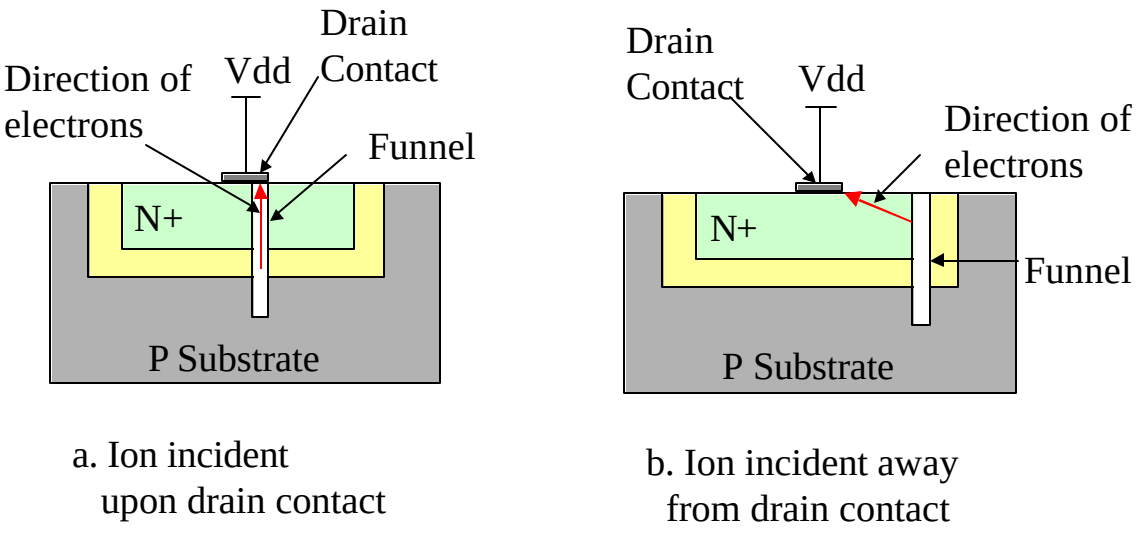
Table 22 also shows that the predicted onset LET is slightly higher than the measured onset LET. This illustrates the sensitivity of the prediction to the funnel length value. It also appears that the predicted cross-sections are greatly overestimated at low LETs. This shows that modeling the sensitive cross-section of a transistor as the drain area surrounded by one depletion width is too large for LET values near the onset threshold. One possible explanation for this is that the charge-collection process is hindered as the charge has to travel farther to get to the drain contact. Figure 50 shows the cross-section of two sensitive NFET drains. In 50a, the ion is incident directly upon

the drain contact. This creates a funnel directly below the contact (shown in white). The red arrow indicates the flow of electrons to the drain. The entire path for an electron is within the funnel, which has extremely high conductivity during the charge-collection process. In contrast, the ion strike and funnel formation are away from the drain contact (but still in the sensitive region) in Figure 50b. In this case, the electrons must travel some distance through the depletion region, where the conductivity is much lower than the conductivity of the funnel. This reduced conductivity reduces the charge-collection efficiency. For low-LET ions, the reduced charge-collection efficiency prevents enough charge to be collected at the drain contact to cause an upset. Larger-LET ions have enough charge that the charge collected at the drain contact is enough, in spite of the reduced collection efficiency.

To support this hypothesis, the prediction for the effective cross-section of Test Program #3 was modified using the drain contact area as the sensitive area for the transistor. Table 23 compares the measured effective cross-section for Test Program #3 with the predicted effective cross-section using the drain contact area as the cross-section and the predicted effective cross-section using the drain area plus depletion width. This comparison shows that the using the drain contact area matches much more closely to the measured value for the cross-section. This indicates that this may be a viable explanation.

Program	Onset LET (MeV*cm^2/mg)	Saturated Cross-Section (cm^2/device)
Measured Test Program #1	< 20.7	3.23e-6 +/- 2.11e-6
Predicted Test Program #1	8.4	3.59e-6
Measured Test Program #2	< 5.69	8.20e-5 +/- 8.44e-6
Predicted Test Program #2	8.4	7.77e-5
Measured Test Program #3	< 5.69	8.9e-5 +/- 1.49e-5
Predicted Test Program #3	8.4	9.16e-5

Table 22. Measured and Predicted Onset LET and Saturated Cross-Section



a. Ion incident upon drain contact

b. Ion incident away from drain contact

Figure 50. Onset-LET Cross-Section Reduction

Measured Cross-section @ Onset LET (cm^2)	Predicted Effective Cross-section @ Onset LET using Active Contact Area (cm^2)	Predicted Effective Cross-section @ Onset LET using Drain Area (cm^2)
1.55e-7	1.67e-7	5.93e-6

Table 23. Measured and Predicted Access-Error Cross-sections for Test Program #3

F. CONCLUSION

The modeling and simulations documented in Chapter IV were used to predict the transitional probabilities of the SET-state-transition model. These probabilities were combined to predict the test-program-dependent effective cross-section of the KDLX processor. The results of the laser testing described in this chapter show very good agreement with the clock-edge effects modeling (which is a key element of the transitional probability $\delta 1$) and the predicted transitional probability $\varepsilon 1$. The results from the heavy-ion testing also show very good agreement between the predicted and measured system-level cross-sections. The combination of the results from the laser testing and the heavy-ion testing validates both the transitional-probability modeling as well as the system-level predictions from Chapter IV. This, in turn validates the modeling approach defined in Chapter III.

VI. CONCLUSION

A. SUMMARY OF RESEARCH

This dissertation formulated, verified and validated a methodology to determine the single-event transient (SET) tolerance of a complex digital system. A 16-bit RISC microprocessor, the KDLX, was the candidate complex digital system.

The formulation of the methodology was based on the SET-state-transition model of Figure 2. State S1 is the error/transient-free state. From this state, three transitional probabilities, $\beta 1$, $\beta 2$, and $\beta 3$ bring the system to states S3 (SEU), S2 (Logic Gate Transient), and S4 (Output Driver Transient), respectively. These transitional probabilities are SET generation probabilities. From state S2, $\delta 1$ is the probability that the transient becomes latched to become an SEU (state S3); $\delta 2$ is the probability that the transient propagates to an output driver (state S4). From state s4, $\varepsilon 2$ is the probability that the transient causes an error to the external system (state S5 - failure). These transitional probabilities are SET propagation probabilities. From state S3, $\varepsilon 1$ is the transitional probability that the SEU will propagate to the output and cause an error to the external system (also state S5). This is the SEU propagation probability.

Determination of the SET generation probabilities is based on three key parameters: critical charge, funnel length, and cross-section. For transitional probability $\beta 1$, the critical charge is the minimum quantity of charge that causes an SEU to the memory element. For $\beta 2$ and $\beta 3$, the critical charge is the minimum quantity of charge that causes a given amplitude and pulsewidth for an SET. It is determined with SPICE simulations by injecting a double-exponential current pulse onto the node of interest and observing the results. The funnel length is the linear distance of charge collection in the ion track. It is a necessary parameter to convert the critical charge to an equivalent linear energy transfer (LET). The cross-section of a transistor is the sensitive region that the ion must hit to cause the SET. It is determined from the layout and the depletion width.

Determination of the SET propagation probabilities is based on SET analog propagation, SET logic propagation, and clock-edge effects modeling. SET analog

propagation modeling determines if a transient has enough energy to propagate, given a sensitized logic path. It is modeled in SPICE. SET logic propagation modeling determines the probability of a sensitized logic path. Clock-edge effects modeling determines the probability that a transient is latched into a memory element. It is accomplished using SPICE simulations.

The probability of SEU propagation is determined by instruction-based register-usage analysis and VHDL-based fault injection. Instruction-based register-usage analysis is top-level analysis to determine which registers are used in a particular functional mode. VHDL-based fault injection is used when the top-level analysis does not provide enough insight into the propagation of SEUs in the design.

To verify this methodology, the SET tolerance of a candidate complex digital system was determined. The candidate system was the 16-bit KDLX RISC processor. This processor was implemented using the MOSIS prototyping service. Using the design information available, the SET generation modeling, SET propagation modeling, and SEU propagation modeling were performed.

The funnel length was estimated to be 3.9 μm based on the 3-dimensional semiconductor simulation results in Dodd[28]. For $\beta 1$, the total effective saturated cross-section of the DFFC standard-cell (the only type of memory element used in the KDLX) was 33.66 μm. The critical charge was 339 fC, or an ion with an LET equal to 8.4 MeV*cm^2/mg. The effective cross-section of the transistor that defined the critical charge was 2.18 μm. For $\beta 2$, the sensitive effective cross-section of the inverter was determined to be 35.79 μm. The critical charge to provide a 3V amplitude, 190 ps length of the SET pulse was determined to be 423.7 fC. This was incident on the NFET and requires an ion with an LET equal to 10.5 MeV*cm^2/mg. For $\beta 3$, it was determined that the capacitance at the output of the KDLX is so large that an ion with an LET greater than 340 MeV*cm2/mg was necessary to provide a small effect at the output. Ions with LETs that large do not exist, thus $\beta 3$ was set to 0.

The SET propagation modeling was necessary to determine $\delta 1$, $\delta 2$ and $\varepsilon 2$ transitional probabilities. The analog propagation was modeled using SPICE. The

modeling showed that there is a propagation threshold. If the amplitude and pulsewidth of the SET are below the threshold, propagation without attenuation will not occur. For inverters, this threshold was determined to be an amplitude of 3.0V and pulsewidth of 400 picoseconds for a 0-1-0 SET and 3.3V, 460 picoseconds for a 1-0-1 SET. For SET logic propagation, it was determined that the multiplexer was critical in determining the logic cross-section. This was because the logic datapath is largely determined by the how the instruction being executeded controls the mulitplexers to steer the data. The clock-edge effects modeling revealed that an SET must have equal amplitude and longer pulse than is required for the propagation threshold (480 picoseconds vs. 400 picoseconds for a 1-0-1 SET). This simplified the SET propagation modeling, because if an SET was large enough (both in amplitude and pulsewidth) to be latched, then it must also be large enough to propagate without attenuation.

The SEU propagation modeling focused on determining the sensitive window for an SEU to occur during the execution of an instruction to cause improper execution. For most registers, this was accomplished by performing instruction-based register-usage analysis for each instruction. The sensitive windows for the pipeline registers were not apparent from this analysis. As a result, VHDL-based fault injection was used to determine the sensitive windows in the pipeline.

The results of the modeling were combined to provide a system-level prediction for the KDLX processor for three different test programs. For Test Program #1, the predicted effective saturated cross-section was 767.55 μm^2. For Test Program #2 the predicted effective saturated cross-section was 6841.22 μm^2, and for Test Program #3, it was 8226.06 μm^2.

Laser testing and heavy-ion testing were performed to validate the results of the modeling. Laser testing was used to validate key transitional probabilities. Heavy-ion testing was performed to validate the system-level predictions.

The laser testing allowed direct injection into states S2 and S3. This way, the predicted transitional probabilities $\delta 1$ and $\varepsilon 1$ could be validated. The logic propagation and clock-edge effects components of $\delta 1$ were validated. To validate logic propagation,

it was shown that a logic gate that is not in the datapath defined by the instruction will not contribute to the cross-section of the instruction. To validate the clock-edge effects modeling, it was shown that the probability of an SET being latched is a linear function of the clock frequency. It was also shown that a longer SET pulse results in a higher probability of being latched. To validate $\varepsilon 1$, the sensitive window concept was tested. For Test Program #1, the predicted $\varepsilon 1$(R1, bit 1) was 0.00391; the measured value was 0.00397. For Test Program #2, the predicted $\varepsilon 1$(R1, Bit 1) was 0.922, and the measured value was 0.931.

The heavy-ion testing validated the system-level predictions. The testing used six different LET values: 5.69, 8.08, 20.6, 28.9, 40.9, and 57.8 MeV*cm^2/mg. In all cases, the cross-section versus LET curves tracked the predicted values for each test program. This was particularly true at LETs > 20 MeV*cm^2/mg. The predicted onset LET was 8.4 MeV*cm^2/mg; the measured was < 5.69 MeV*cm^2/mg (this was the lowest LET tested). The predicted saturated cross-section of Test Program #1 was 3.59e-6 cm^2/device; the measured value was 3.23e-6 cm^2/device. For Test Program #2, the predicted saturated cross-section was 7.77e-5 cm2/device; the measured value was 8.20e-5 cm^2/device. For Test Program #3, the predicted saturated cross-section was 9.16e-5 cm2/device; the measured value was 8.9e-5 cm^2/device.

B. THE 90% SOLUTION

The methodology implemented in this dissertation demonstrates very good agreement between predicted and measured values. A closer look at these results reveals that the system-level cross-section is dominated by the S1-to-S3-to-S5 transition path. This is largely due to the fact that the clock-edge effects minimize the $\delta 1$ transitional probability.

The 90% solution uses this fact to simplify the methodology. All modeling and simulations can focus on determining the transitional probabilities of the S1-to-S3-to-S5 transition path. The problem then reduces to determining the cross-section versus LET curve of the memory elements ($\beta 1$) and determining the probability of SEU propagation

($\varepsilon 1$). The transitional probability $\beta 1$ is determined using SPICE simulations and $\varepsilon 1$ is determined using the instruction–based register-usage analysis.

C. ORIGINAL CONTRIBUTIONS

The primary original contribution of this dissertation is the methodology to determine the effective cross-section of a complex digital system. In particular, the use of the SET-state-transition diagram is unique. This allows the combinational-logic contribution of the cross-section to be determined separately from the static cross-section.

The instruction-based register-usage analysis approach to determine the probability of SEU propagation is also unique, because it provides a precise measure of the sensitive window for a register as a function of the instruction execution. This allows the total number of modes of a processor to be reduced to something that is workable to determine the cross-section for each functional mode.

The SPICE injection circuits are also contributions, because they allow for more accurate SPICE simulations of SETs. While these injection circuits do not provide the accuracy of a high-end mixed-mode simulation that uses 3-dimensional semiconductor modeling, they do represent a significant improvement over the independent current pulse injection approach. This improvement is due to the fact that the charge collected is a function of the injection node voltage with the injection circuits used in this dissertation, while the charge collected using the independent current pulse is not a function of the node voltage.

D. EXTENSION TO OTHER IMPLEMENTATIONS

This dissertation documented the application of the methodology to the KDLX RISC processor. It was validated with the laser and heavy-ion testing, which is important, but for this dissertation to be complete, how the methodology can be applied to other classes of complex digital systems must also be shown.

1. The Standard-Cell Application Specific Integrated Circuit (ASIC) [4]

The first alternate implementation considered is the standard-cell ASIC. Because this is a standard-cell design, the designer has the same information available from the

parametric test results of the foundry run, the extracted layout, and the gate-level hardware-description language (HDL) definition of the design. Thus, the approach is the same as the KDLX to determine the SET generation probabilities. The analog propagation and clock-edge effects modeling are also the same. The difference between the standard-cell ASIC and the KDLX processor is that the ASIC does not have an instruction set. This means that the instruction-based register-usage analysis approach does not apply. Thus, instead of determining the datapath and the ε1 transitional probability for each instruction, these must evaluated for each functional mode. The total effective device cross-section is then determined by the equation:

$$\sigma_{device} = \Sigma\ s_n * D_n\ ,\ n = 1 \text{ to the total number of modes, where} \quad (6.1)$$

s_n = mode-dependent cross-section for mode n,

D_n = duty cycle of state n.

2. Field Programmable Gate Array (FPGA)

The next alternate implementation considered is an FPGA. This is fundamentally different from the standard-cell ASIC because the engineer typically will not have the parametric test results of the foundry run or a SPICE transistor-level model of the logic modules. However, the designer will have a high-level description of the design, as well as a synthesized logic-module description. It is also likely that the engineer has some SEU data on the FPGA logic modules [47]. Assuming that there is SEU test data on the logic modules, the problem becomes determining the functional-mode-dependent cross-section. This requires a determination of the number of logic modules used for each functional mode. The total device cross-section is then determined using equation 6.1.

3. Off-the-Shelf Processor

An off-the-shelf processor is the next implementation considered. In this case, the only information likely to be available is an instruction-set architecture (ISA) description of the processor. The information contained in an ISA description may be similar to the information in a full HDL description of the microarchitecture, except that the ISA

[4] While it is true that an ASIC can be a processor, here an ASIC is defined as a complex digital system that does not contain an instruction set.

description will be missing the hidden memory elements that are included in the HDL description. Examples of hidden memory elements in the KDLX are the pipeline memory elements, the ALU-buffer memory elements, and the data input/output buffer memory elements. These hidden memory elements may create a very big difference between the ISA description and the HDL microarchitecture description. These hidden elements contribute to the effective cross-section, but are not apparent in looking at the ISA description of the processor.

The approach for the processor focuses on determining the instruction-dependent cross-section using a combination of what is known from the ISA description and what can be determined about the microarchitecture from SEU testing. The procedure is as follows:

1. For each instruction type, determine the sensitive memory elements using the ISA description.

2. Add a variable to depict the additional hidden sensitive memory elements for each instruction to the result in #1.

3. Create a test program for each instruction type to determine the mode-dependent cross-section associated with that instruction type. Predict the cross-section of the test program using #1 and #2.

4. Create a (or use an existing) program that uses many different instructions. This is the validation program. Predict the cross-section using #1 and #2.

5. At a high LET (to insure the best statistics), run all programs to determine the saturated cross-sections for each program.

6. Using the test program with the largest saturated cross-section, test the processor at lower LETs to obtain a cross-section versus LET curve. The other test programs should follow this same cross-section versus LET trend.

7. Determine the hidden number of memory elements for each instruction by comparing the resulting saturated cross-sections with the cross-sections that were predicted in #3.

8. Update the instruction-dependent cross-sections by including the contribution of the hidden memory elements.

9. Compare the measured cross-section of the validation program to the updated predicted cross-section. If the two agree, the predicted instruction-dependent cross-sections will be validated.

4. Off-the-Shelf ASIC

The final implementation considered is an off-the-shelf ASIC. In this case, the device is not a processor, and the only information typically available is a block diagram from the data sheet, which may be significantly different from the actual architecture. As before, it is necessary to determine the cross-section for each of the operational modes of the device. Each functional mode of the device should be tested at a high LET to determine the relative cross-section of each mode. Then, as with the off-the-shelf processor, the cross-section versus LET curve is determined by testing at lower LETs with the device operating in the functional mode with the largest cross-section. The total effective device cross-section is determined by using equation 6.1.

E. AREAS FOR FURTHER INVESTIGATION

There are two primary areas for further investigation. The first area is the verification of the approaches defined in the previous section. This could be accomplished by implementing the approach on a standard-cell ASIC, FPGA, off-the-shelf processor, and off-the-shelf ASIC. This would demonstrate the versatility of the methodology defined in this dissertation.

The second area is the determination of how the sensitive area of a transistor is reduced as the LET approaches the onset LET. In this dissertation, it was assumed that the sensitive area is zero if the LET is less than the onset LET and becomes the area defined by equations 3.12 – 3.14 once the onset LET has been reached. At high LETs, the test results in Chapter V show that this estimate is accurate; however at LETs less than 10 MeV*cm^2/mg, the measured cross-section is an order of magnitude less than the predicted cross-section. This indicates that the assumption is not valid at lower LETs, which is why further investigation is merited. The recommended approach is to perform mixed-mode 3-dimensional simulations of the transistors in the DFFC standard-cell flip-

flop. These simulations allow the precise control of the location that the incident ion strikes the sensitive area. This would show if the charge-collection process is enhanced as the location of the funnel moves closer to the drain contact (see Figure 50). The simulation results could be validated with testing using heavy ions with low values of LET.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A – KDLX PROCESSOR DESCRIPTION

A. INTRODUCTION

The KDLX processor is a 16-bit version of the DLX processor described in Computer Architecture, a Quantitative Approach, by Hennessey and Patterson[48]. It was implemented through the MOSIS prototyping service using the Hewlett-Packard 0.5 µm CMOS process. The processor was designed using the Tanner Tools Pro MOSIS SCMOS Standard-cell library, with a gate length of 0.7 µm. This appendix describes the functional blocks of the processor as well as the instruction set that was implemented. Figure 51 shows a photograph of the device. Figure 52 shows the layout of the KDLX.

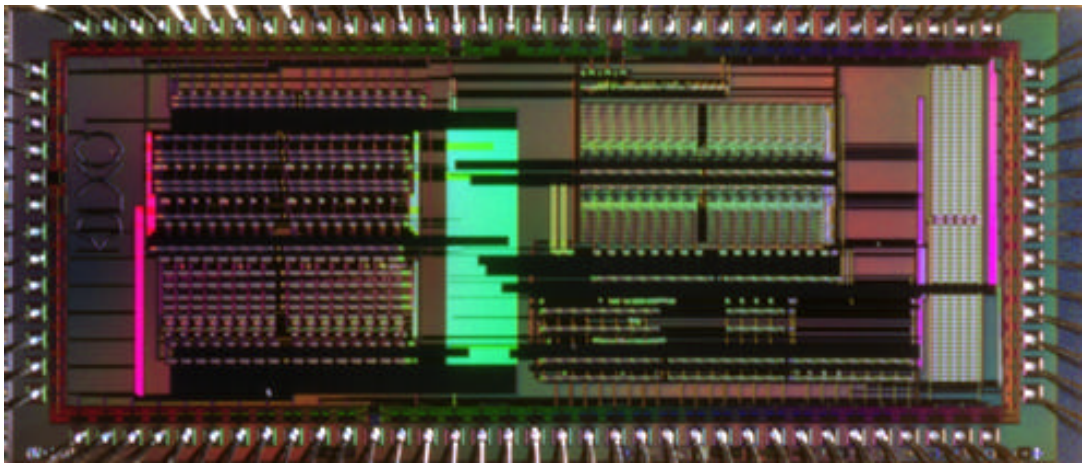


Figure 51. Photograph of KDLX Processor

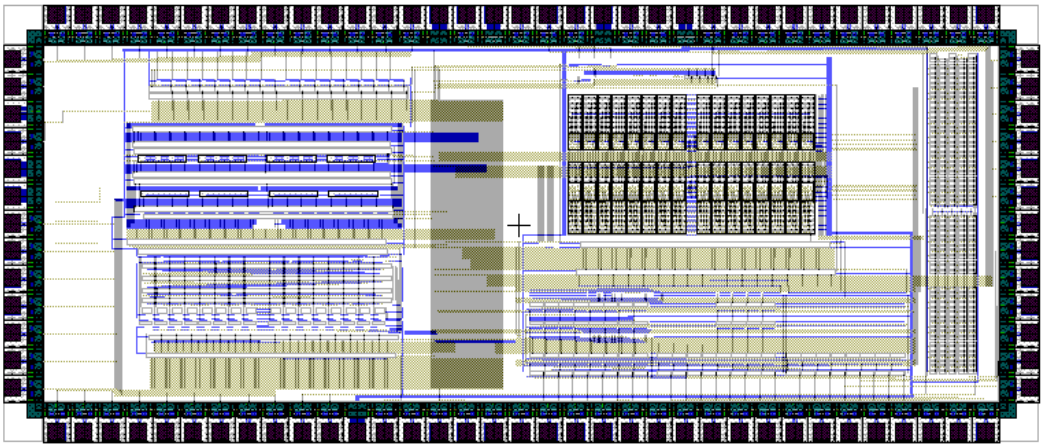


Figure 52. KDLX Layout

B. TOP-LEVEL FUNCTIONAL BLOCK DIAGRAM

Figure 53 shows the top-level block diagram of the KDLX processor. It consists of the following functional blocks:

1. General-Purpose Register File – contains the general-purpose 16-bit registers (R0 – R16).

2. ALU (Arithmetic Logic Unit) – performs arithmetic and logical functions on its inputs.

3. PC Control Module – controls the program counter.

4. Pipeline Module – implements the pipeline by providing the necessary control signals to the other modules during each pipeline stage.

5. RW_Control – provides the Rd* and Wr* signals to control the input and output of the processor.

6. ALU Out Buffer – buffers the output of the ALU to drive the ADDR_Out signal and to feedback into the Delayed_ALU_Out_Buffer Register for writeback.

7. Delayed ALU Out Buffer – buffers the output of the ALU for writeback into the general-purpose registers.

8. Data Out Buffer – buffers the output data when writing to memory.

9. Data In Buffer – buffers the input data when reading from memory.

10. Reg_In_Sel Multiplexer – selects the input to the general-purpose registers. This input can be the Program_Addr+2 signal, the Data_In register, or the Delayed_ALU_Out Buffer (from the ALU).

11. A_Mux – selects the A input for the ALU. This input can be the Program_Addr+1(15:0) from the PC Control Module, the 16-bit immediate (Immed[15:0]), the high 8-bit immediate (Immed[7:0]), or the register RA from the general-purpose register file.

12. B_Mux – selects the B input for the ALU. This can the sign-extended immediate (S_Immed), which is the sign-extended 8-bit immediate value, the

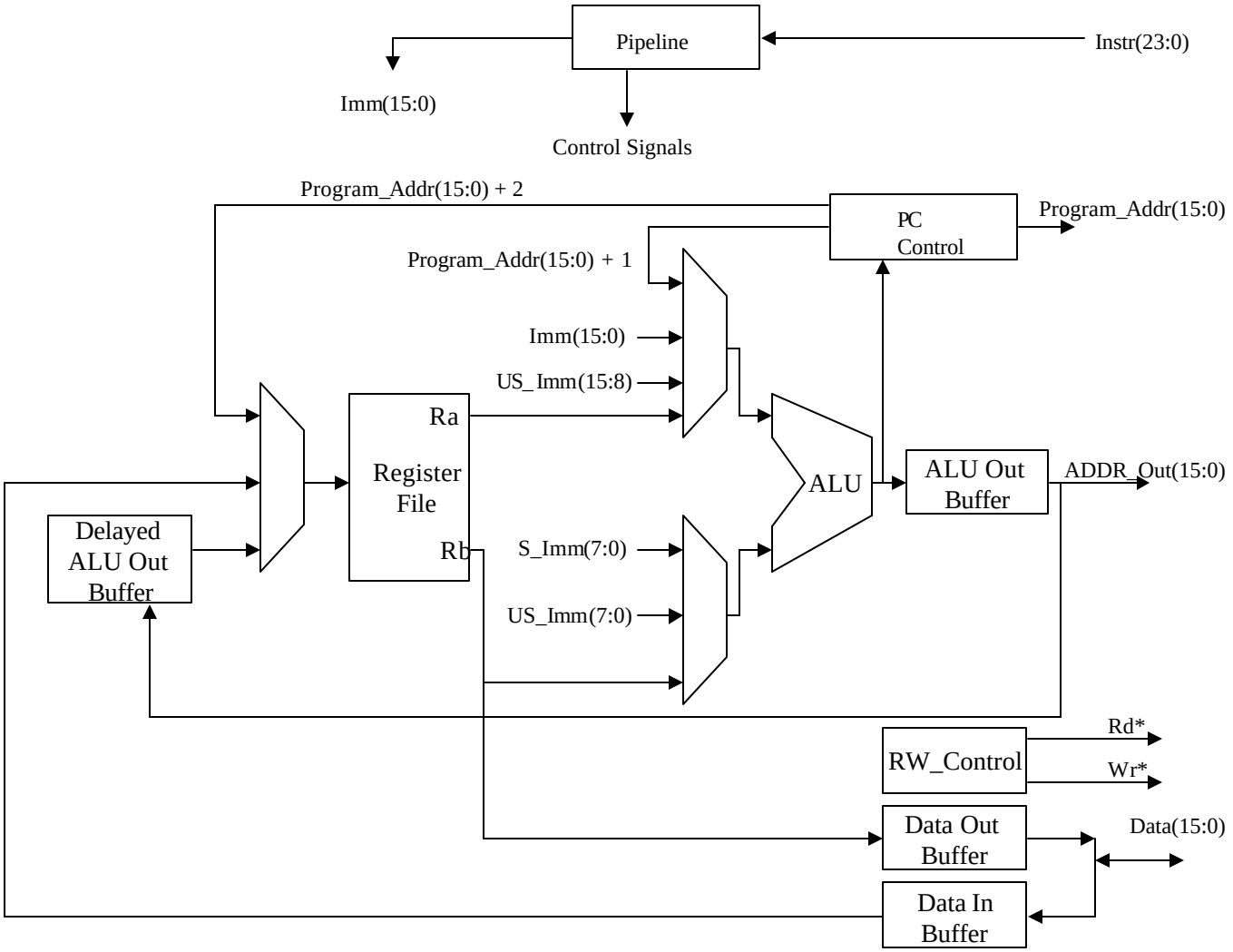


Figure 53. KDLX Block Diagram

unsigned-immediate (US_Immed) value, or the register RB from the general-purpose register file.

C. GENERAL-PURPOSE REGISTER FILE

Figure 54 shows the block diagram of the general-purpose register file. The general-purpose register selected by the Dest(3:0) signal is written through the Reg_Data_In input when the WB_Enable signal is active. The registers RA and RB are loaded with the registers selected by the RS1(3:0) and RS2(3:0) inputs.

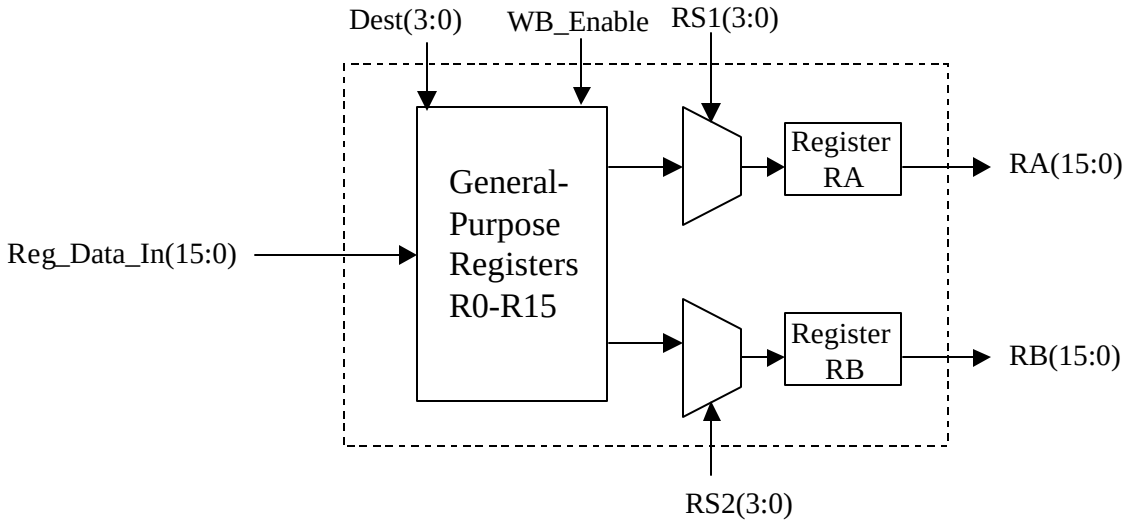


Figure 54. General-Purpose Register File Block Diagram

D. PIPELINE MODULE

Figure 55 shows a block diagram of the Pipeline Module. The Fetch pipeline cycle is not shown because it fetches the 24-bit instruction word, Instr(23:0). The Decode stage provides the RS1 and RS2 multiplexer selections for the General-Purpose Register File. The Execute stage is the stage in which the ALU operations are performed. In this stage, the pipeline module provides the 8 or 16-bit immediate value (depending on the instruction type). The ALU_Op(4:0) defines the operation of the ALU module. The A_Mux and B_Mux signals are used to define the input to the ALU. The PC_Sel controls the source of the next program counter for the PC Control module. The Memory Stage provides the Rd_Enable and Wr_Enable to the RW_Control module to drive the Rd* and Wr* outputs. The Writeback stage provides the WB_Enable, Dest(3:0), and

Reg_In_Sel(1:0) to the General-Purpose Register File. Additionally, the Writeback stage also provides the Interrupt Address Register Enable (IAR_Enable) to the PC Control module to save the return address during a TRAP instruction.

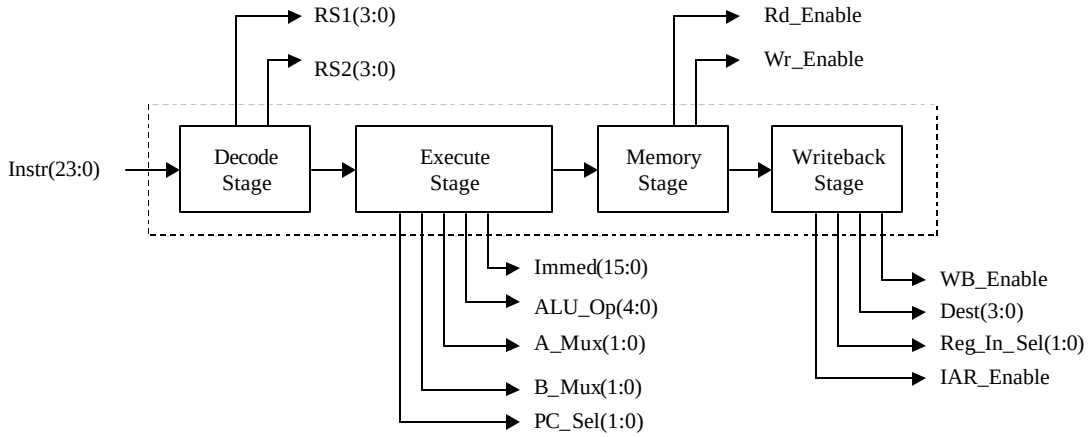


Figure 55. Pipeline Module Block Diagram

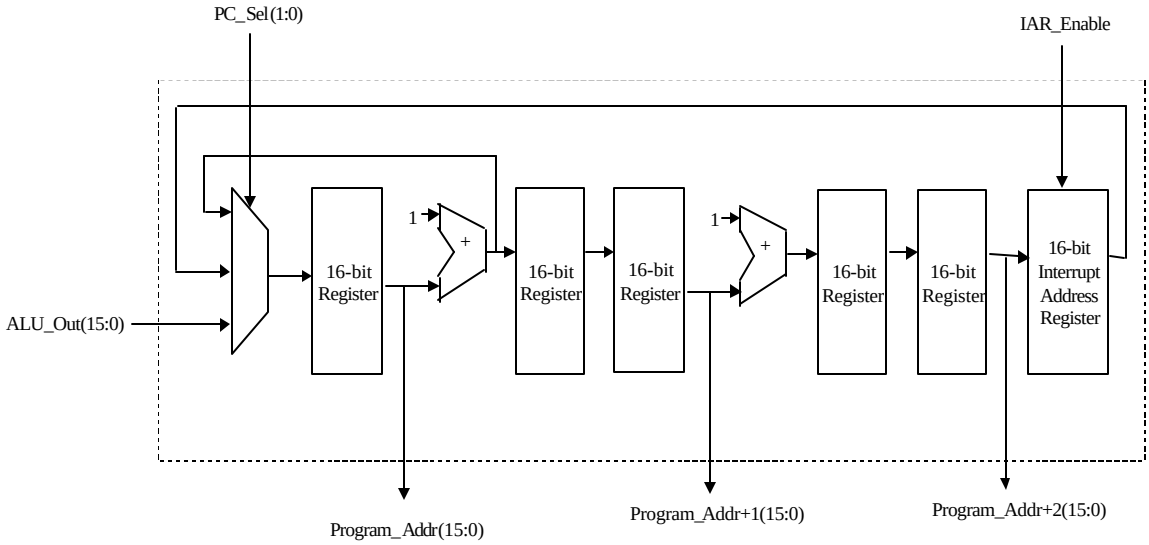


Figure 56. PC Control Module Block Diagram

E. PC CONTROL MODULE DESCRIPTION

Figure 56 shows a block diagram of the PC Control module. The PC_Sel(1:0) selects the source for the next program address. The sources can be the incremented Program Address (normal operation), the Interrupt Address(for a return from exception, or RFE, instruction) or the output of the ALU, ALU_Out (for Jump and Branch instructions). The PC Control module provides the Program_Addr(15:0) output to

perform the instruction fetch. The Program_Addr+1(15:0) output is used by the ALU to determine the next program address for Branch instructions. The Program_Addr+2(15:0) is the link address used in the jump and link instructions.

F. ARITHMETIC LOGIC UNIT (ALU) MODULE

Figure 57 shows a block diagram of the ALU Module. The adder module performs addition and subtraction. The ALU logic module provides the capability to perform the logical functions AND, OR, and exclusive-OR. The barrel shifter is used for bit-wise shift operations. The conditional set module is used for the set-on conditional functions.

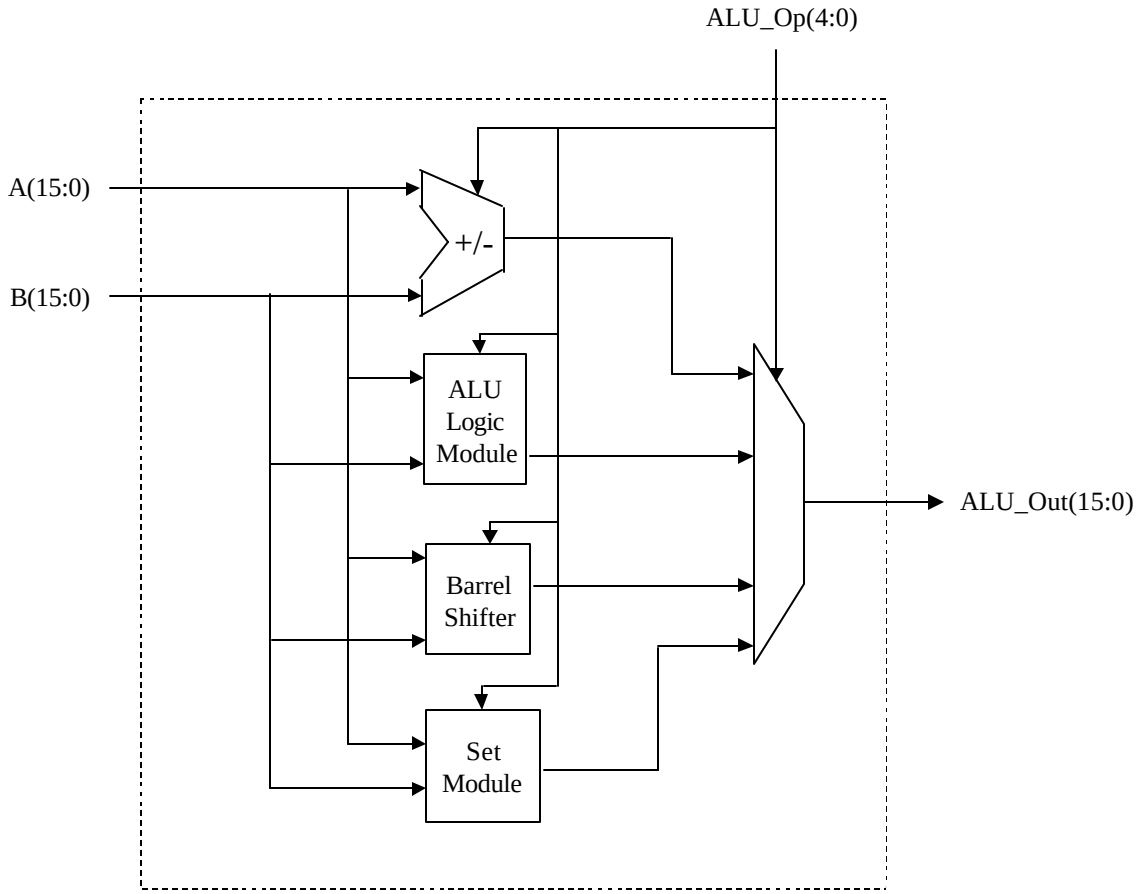
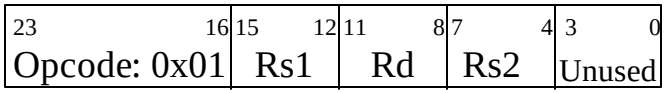


Figure 57. Arithmetic Logic Unit (ALU) Module Block Diagram

G. INSTRUCTION SET DESCRIPTION

This section describes the instructions that were implemented in the KDLX. The format of the instructions is based on the description in Sailer[49].

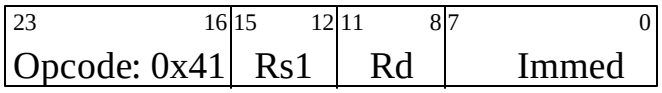
Instruction: ADD (Register Add)



Usage: ADD Rd, Rs1, Rs2

Operation: (Rs1 + Rs2) = Rd

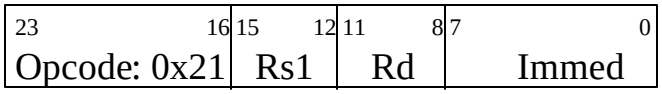
Instruction: ADDI (Add Immediate)



Usage: ADD Rd, Rs1, Rs2

Operation: (Rs1 + $[(\text{Immed}_7)^8$ ||Immed])= Rd

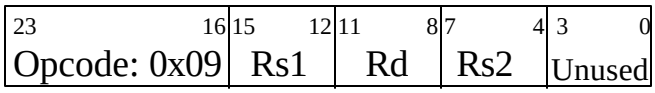
Instruction: ADDUI (Add Unsigned Immediate)



Usage: ADD Rd, Rs1, Rs2

Operation: (Rs1 + $[(0)^8$ ||Immed])= Rd

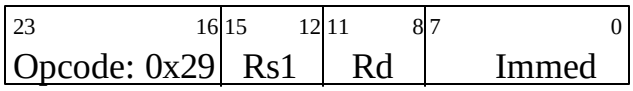
Instruction: AND (Register AND)



Usage: AND Rd, Rs1, Rs2

Operation: (Rs1 (logical-and) Rs2) = Rd

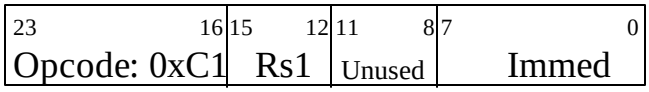
Instruction: ANDI (And Immediate)



Usage: AND Rd, Rs1, Immed

Operation: (Rs1 logical-and $[(Immed_7)^8 || Immed]$)= Rd

Instruction: BEQZ (Branch if Equal to Zero)

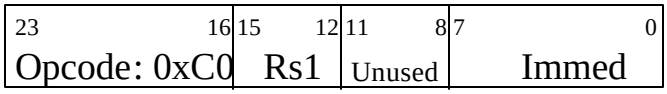


Usage: BEQZ Rs1, Immed

Operation: If Rs1 = 0,

then Program_Addr = $(PC+1 + [(Immed_7)^8 || Immed])$

Instruction: BNEZ (Branch if Not Equal to Zero)



Usage: BNEZ Rs1, Immed

Operation: If Rs1 != 0,

then Program_Addr = $(PC+1 + [(Immed_7)^8 || Immed])$

Instruction: J (Jump)

Usage: J Immed

Operation: Program_Addr = Immed

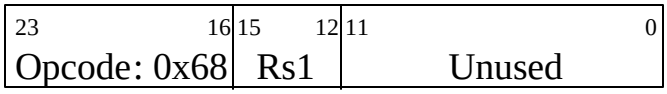
Instruction: JAL (Jump and Link)

Usage: JAL Immed

Operation: Program_Addr = Immed;

R15 = Link_Program_Address

Instruction: JALR (Jump Register and Link)

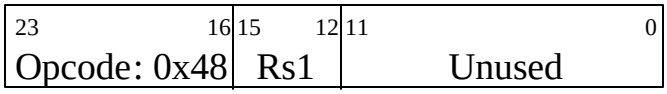


Usage: JALR Rs1

Operation: Program_Addr = (Rs1)

R15 = Link_Program_Address

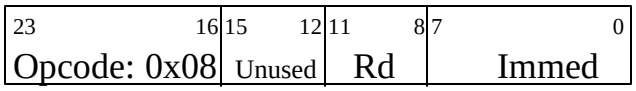
Instruction: JR (Jump Register)



Usage: JALR Rs1

Operation: Program_Addr = (Rs1)

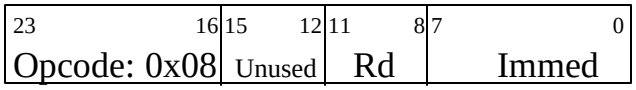
Instruction: LHI (Load High Immediate)



Usage: LHI Rd, Immed

Operation: Rd = Immed || $(0)^8$

Instruction: LW (Load Word)



Usage: LW Rd, Rs1(Immed)

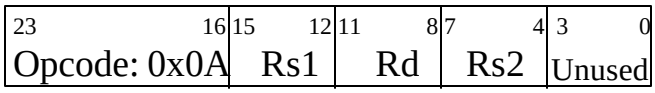
Operation: Rd = Mem{Rs1 + [$(\text{Immed}_7)^8$ ||Immed]}

Instruction: NOP (No Operation)

Usage: NOP

Operation: None

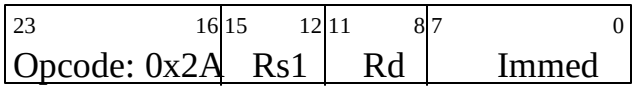
Instruction: OR (Register OR)



Usage: OR Rd, Rs1, Rs2

Operation: Rd = (Rs1 (logical-or) Rs2)

Instruction: ORI (OR Immediate)



Usage: OR Rd, Rs1, Immed

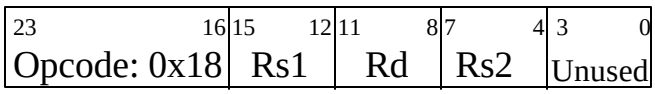
Operation: (Rs1 logical-or $[(Immed_7)^8$ ||Immed])= Rd

Instruction: RFE (Return from Exception)

Usage: RFE

Operation: Program_Address = Interrupt_Address_Register

Instruction: SEQ (Set Equal)

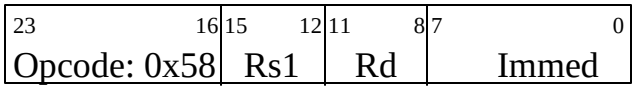


Usage: SEQ Rd, Rs1, Rs2

Operation: If Rs1 = Rs2, then Rd = 0x0001,

Else, Rd = 0x0000

Instruction: SEQI (Set Equal Immediate)

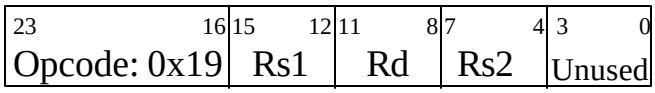


Usage: SEQI Rd, Rs1, Immed

Operation: If Rs1 = $[(Immed_7)^8$ ||Immed]), then Rd = 0x0001,

Else, Rd = 0x0000

Instruction: SGE (Set if Greater Than or Equal)

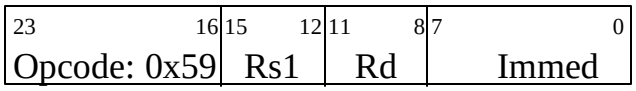


Usage: SGE Rd, Rs1, Rs2

Operation: If Rs1 >= Rs2, then Rd = 0x0001,

Else, Rd = 0x0000

Instruction: SGEI (Set if Greater Than or Equal Immediate)

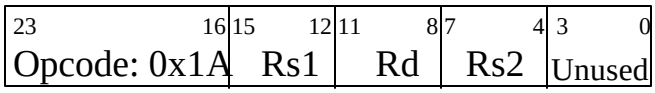


Usage: SGEI Rd, Rs1, Immed

Operation: If Rs1 >= [$(Immed_7)^8$ ||Immed]), then Rd = 0x0001,

Else, Rd = 0x0000

Instruction: SGT (Set if Greater Than)

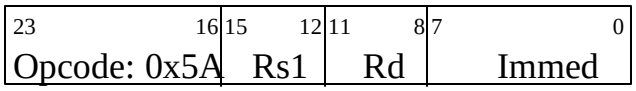


Usage: SGT Rd, Rs1, Rs2

Operation: If Rs1 > Rs2, then Rd = 0x0001,

Else, Rd = 0x0000

Instruction: SGTI (Set if Greater Than Immediate)

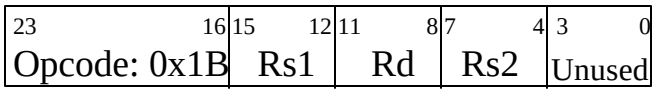


Usage: SGTI Rd, Rs1, Immed

Operation: If Rs1 > [$(Immed_7)^8$ ||Immed]), then Rd = 0x0001,

Else, Rd = 0x0000

Instruction: SLE (Set if Less Than or Equal)

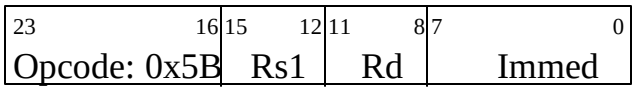


Usage: SLE Rd, Rs1, Rs2

Operation: If Rs1 < Rs2, then Rd = 0x0001,

Else, Rd = 0x0000

Instruction: SLEI (Set if Less Than or Equal Immediate)

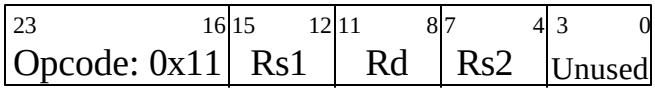


Usage: SLEI Rd, Rs1, Immed

Operation: If Rs1 <= [$(Immed_7)^8$ ||Immed]), then Rd = 0x0001,

Else, Rd = 0x0000

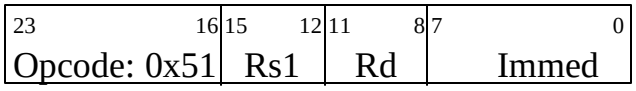
Instruction: SLL (Shift Logic Left)



Usage: SLL Rd, Rs1, Rs2

Operation: Rd = (Rs1) shifted left by Rs2(3:0) bits

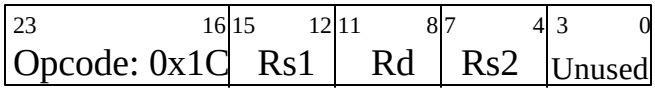
Instruction: SLLI (Shift Logic Left Immediate)



Usage: SLLI Rd, Rs1, Immed

Operation: Rd = (Rs1) shifted left by Immed(3:0) bits

Instruction: SLT(Set if Less Than)

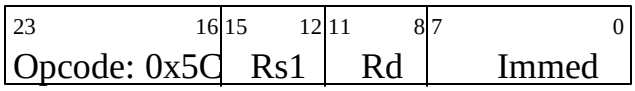


Usage: SLT Rd, Rs1, Rs2

Operation: If Rs1 < Rs2, then Rd = 0x0001,

Else, Rd = 0x0000

Instruction: SLTI (Set if Less Than Immediate)

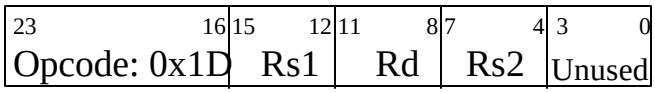


Usage: SLTI Rd, Rs1, Immed

Operation: If Rs1 < [$(Immed_7)^8$ ||Immed]), then Rd = 0x0001,

Else, Rd = 0x0000

Instruction: SNE (Set If Not Equal)

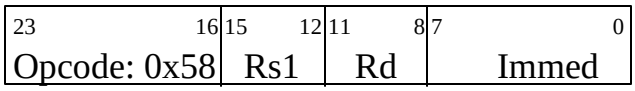


Usage: SNE Rd, Rs1, Rs2

Operation: If Rs1 != Rs2, then Rd = 0x0001,

Else, Rd = 0x0000

Instruction: SNEI (Set If Not Equal Immediate)

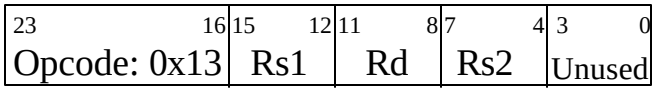


Usage: SNEI Rd, Rs1, Immed

Operation: If Rs1 != [$(Immed_7)^8$ ||Immed]), then Rd = 0x0001,

Else, Rd = 0x0000

Instruction: SRA (Shift Right Arithmetic)

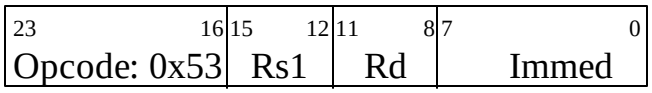


Usage: SRA Rd, Rs1, Rs2

Operation: Rd = (Rs1) shifted right by Rs2(3:0) bits,

with Rs1(15) shifted in from right (for sign extension)

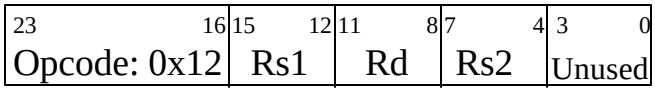
Instruction: SRAI (Shift Right Arithmetic Immediate)



Usage: SRAI Rd, Rs1, Immed

Operation: Rd = Rd = (Rs1) shifted right by Immed(3:0) bits, with Rs1(15) shifted in from right (for sign extension)

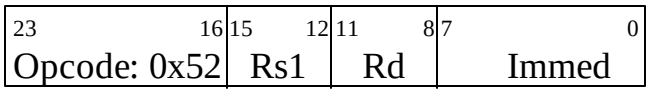
Instruction: SRL (Shift Right Logical)



Usage: SRL Rd, Rs1, Rs2

Operation: Rd = (Rs1) shifted left by Rs2(3:0) bits, with 0s shifted in from right

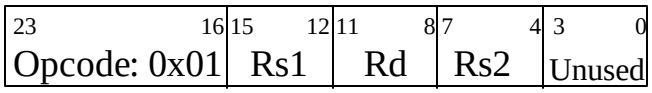
Instruction: SRLI (Shift Right Logical Immediate)



Usage: SRLI Rd, Rs1, Immed

Operation: Rd = (Rs1) shifted left by Rs2(3:0) bits, with 0s shifted in from right

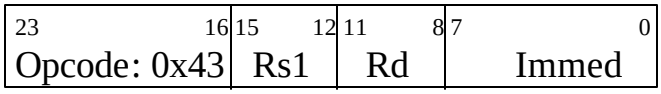
Instruction: SUB (Register Subtract)



Usage: Sub Rd, Rs1, Rs2

Operation: Rd = (Rs1 - Rs2)

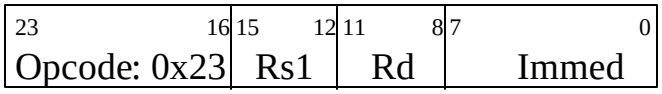
Instruction: SUBI (Subtract Immediate)



Usage: SUB Rd, Rs1, Rs2

Operation: Rd = (Rs1 - [$(Immed_7)^8$ ||Immed])

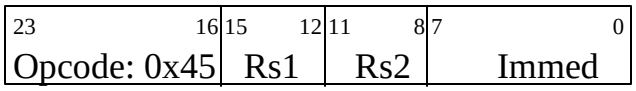
Instruction: SUBUI (Subtract Unsigned Immediate)



Usage: Sub Rd, Rs1, Rs2

Operation: Rd = (Rs1 - $[(0)^8$ ||Immed])

Instruction: SW (Store Word)



Usage: SW Rs2, Rs1(Immed)

Operation: Mem{Rs1 + $[(\text{Immed}_7)^8$ ||Immed]} = Rs2

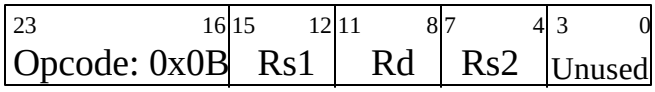
Instruction: TRAP (Software Trap)

Usage: Trap Immed

Operation: Program_Addr = Immed

Interrupt Address Register = Link_Program_Address

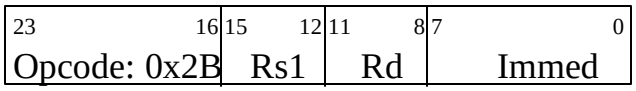
Instruction: XOR (Register Exclusive-OR)



Usage: XOR Rd, Rs1, Rs2

Operation: Rd = (Rs1 (exclusive-or) Rs2)

Instruction: XORI (Exclusive-OR Immediate)



Usage: XORI Rd, Rs1, Immed

Operation: (Rs1 (exclusive-or) $[(\text{Immed}_7)^8$ ||Immed])= Rd

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B – SIMULATION RESULTS

A. OBJECTIVE

The objective of Appendix B is to document the simulation results that were not documented in Chapter IV. These results are divided into four sections: SET generation modeling, SET analog propagation modeling, logic path modeling and Instruction-based register-usage analysis.

B. SET GENERATION MODELING

1. Objective

This objective of this section is to document the simulation results of the SET Generation Modeling. Chapter IV documents the results of the SET Generation modeling on the DFFC standard cell, the INV standard cell, and the output driver. This section documents the remaining standard cells: Nand2, Nand3, Nand4, Nor2, Nor3, Nor4, Xor2, Mux2, and Buf4.

2. Nand2

The Nand2 (two input Nand gate) has four different input possibilities. Figure 58 shows the schematic, and Table 24 shows the cross-sections and LETs for each of the sensitive regions with the resulting amplitude and pulsewidth at the output.

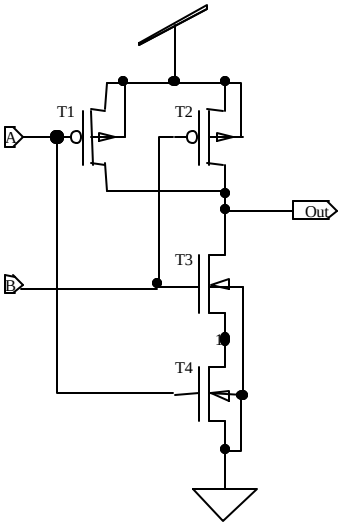


Figure 58. Nand2 Schematic (after [50])

Input State	**Senstive Transistor**	**Charge (fC)**	**LET (MeV*cm^2 /mg)**	**Cross-Section (μm^2)**	**Effective Cross-Section (μm^2)**	**Output Ampli-tude (Volts)**	**Output Pulse-Width (ps)**
A=0 B=0	T5 (NFET)	531	14.32	41.59	10.40	-2.58	130
		791	21.33	41.59	10.40	-3.25	200
		921	24.88	41.59	10.40	-3.3	240
		973	26.24	41.59	10.40	-3.3	250
A=0 B=1	T6 (NFET)	284	7.66	31.76	7.94	-1.36	200
		477	12.86	31.76	7.94	-2.29	250
		556	14.99	31.76	7.94	-2.59	270
		588	15.86	31.76	7.94	-2.69	280
A=1 B=0	T5 (NFET)	402	10.84	41.59	10.40	-2.86	180
		530	14.29	41.59	10.40	-3.27	250
		593	15.99	41.59	10.40	-3.3	280
		618	16.67	41.59	10.40	-3.3	300
A=1 B=1	T11, T12 (PFETs)	479	12.92	42.24	10.56	2.71	150
		644	17.37	42.24	10.56	3.26	220
		720	19.42	42.24	10.56	3.3	260
		755	20.36	42.24	10.56	3.3	270

Table 24. Nand2 Simulation Results

3. Nand3

The Nand3 (three-input Nand gate) has eight different input possibilities. Figure 59 shows the schematic, and Table 25 shows the cross-sections and LETs for each of the sensitive regions with the resulting amplitude and pulsewidth at the output.

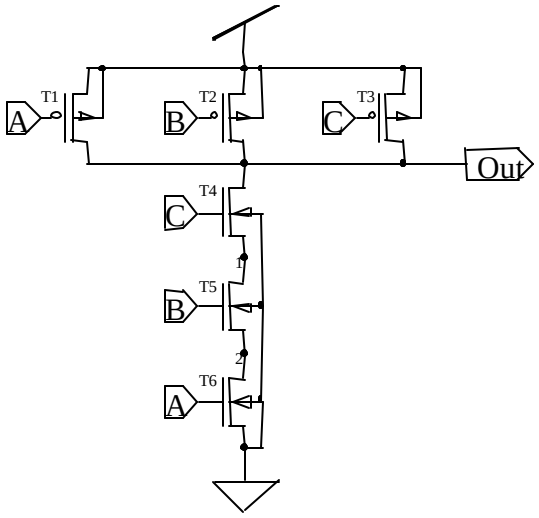


Figure 59. Nand3 Schematic (after [51])

Input State	Senstive Transistor	Charge (fC)	LET (MeV*cm^2/mg)	Cross-Section (mm^2)	Effective Cross-Section (mm^2)	Output Ampli-tude (Volts)	Output Pulse-Width (ps)
A=0 B=0 C=0	T2 (NFET)	750	20.23	41.59	5.20	-2.34	130
		1330	35.87	41.59	5.20	-3.24	210
		1830	49.35	41.59	5.20	-3.3	320
		2200	59.33	41.59	5.20	-3.3	420
A=0 B=0 C=1	T3 (NFET)	334	9.01	31.76	3.97	-0.8	180
		692	18.66	31.76	3.97	-1.4	240
		981	26.46	31.76	3.97	-1.77	330
		1200	32.36	31.76	3.97	-1.94	360
A=0 B=1 C=0	T2 (NFET)	631	17.02	41.59	5.20	-1.36	200
		1020	27.51	41.59	5.20	-2.29	250
		1350	36.41	41.59	5.20	-2.59	270
		1590	42.88	41.59	5.20	-2.69	280
A=0 B=1 C=1	T4 (NFET)	274	7.39	31.76	3.97	-0.83	270
		484	13.05	31.76	3.97	-1.44	330
		663	17.88	31.76	3.97	-1.82	410
		784	21.14	31.76	3.97	-2.02	460
A=1 B=0 C=0	T2 (NFET)	631	17.02	41.59	5.20	-1.36	200
		1020	27.51	41.59	5.20	-2.29	250
		1350	36.41	41.59	5.20	-2.59	270
		1590	42.88	41.59	5.20	-2.69	280
A=1 B=0 C=1	T3 (NFET)	330	8.90	31.76	3.97	-1.29	240
		598	16.13	31.76	3.97	-2.25	330
		774	20.87	31.76	3.97	-2.61	400
		895	24.14	31.76	3.97	-2.71	470
A=1 B=1 C=0	T4 (NFET)	483	13.03	41.59	5.20	-2.84	220
		682	18.39	41.59	5.20	-3.2	310
		844	22.76	41.59	5.20	-3.3	420
		956	25.78	41.59	5.20	-3.3	480
A=1 B=1 C=1	T6, T7, T8 (PFET)	526	14.19	62.71	7.84	2.73	220
		713	19.23	62.71	7.84	3.25	320
		853	23.0	62.71	7.84	3.3	420
		951	25.65	62.71	7.84	3.3	480

Table 25. Nand3 Simulation Results

4. Nand4

The Nand4 (four input Nand gate) has sixteen different input possibilities. Figure 60 shows the schematic, and Tables 26 and 27 show the cross-sections and LETs for each of the sensitive regions with the resulting amplitude and pulsewidth at the output.

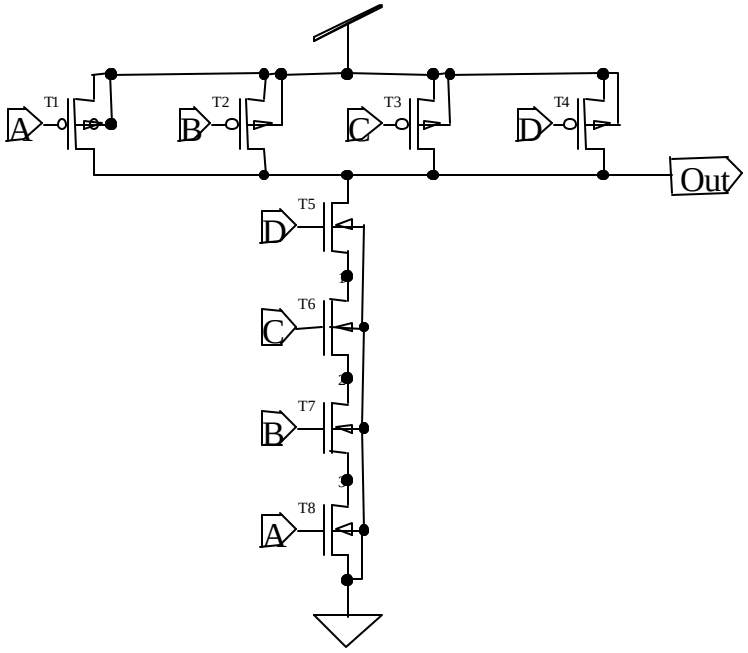


Figure 60. Nand4 Schematic (after [52])

Input State	Senstive Transistor	Charge (fC)	LET (MeV*cm^2/mg)	Cross-Section (mm^2)	Effective Cross-Section (mm^2)	Output Ampli-tude (Volts)	Output Pulse-Width (ps)
A=0 B=0 C=0 D=0	T1 (NFET)	1060	28.57	41.59	2.60	-2.61	130
		1570	42.34	41.59	2.60	-3.28	200
		1820	49.08	41.59	2.60	-3.3	240
		1920	51.78	41.59	2.60	-3.3	250
A=0 B=0 C=0 D=1	T4_1 (NFET)	392	10.57	31.76	1.99	-0.61	180
		624	16.83	31.76	1.99	-0.9	220
		747	20.15	31.76	1.99	-1.04	250
		791	21.33	31.76	1.99	-1.11	260
A=0 B=0 C=1 D=0	T1 (NFET)	932	25.13	41.59	2.60	-2.74	150
		1310	35.33	41.59	2.60	-3.27	220
		1500	40.45	41.59	2.60	-3.3	250
		1570	42.34	41.59	2.60	-3.3	260
A=0 B=0 C=1 D=1	T5 (NFET)	325	8.76	31.76	1.99	-0.5	240
		461	12.43	31.76	1.99	-0.72	260
		533	14.37	31.76	1.99	-0.83	270
		557	15.02	31.76	1.99	-0.87	270
A=0 B=1 C=0 D=0	T1 (NFET)	932	25.13	41.59	2.60	-2.74	150
		1310	35.33	41.59	2.60	-3.27	220
		1500	40.45	41.59	2.60	-3.3	250
		1570	42.34	41.59	2.60	-3.3	260
A=0 B=1 C=0 D=1	T4_1 (NFET)	393	10.60	31.76	1.99	-0.82	200
		623	16.80	31.76	1.99	-1.29	230
		733	19.77	31.76	1.99	-1.55	240
		775	20.90	31.76	1.99	-1.65	250
A=0 B=1 C=1 D=0	T1 (NFET)	796	21.47	41.59	2.60	-2.89	190
		1040	28.05	41.59	2.60	-3.3	240
		1170	31.55	41.59	2.60	-3.3	270
		1220	32.90	41.59	2.60	-3.3	290
A=0 B=1 C=1 D=1	T6 (NFET)	307	8.28	31.76	1.99	-0.53	380
		414	11.17	31.76	1.99	-0.78	370
		462	12.46	31.76	1.99	-0.92	380
		504	13.59	31.76	1.99	-0.92	380

Table 26. Nand4 SET Generation Modeling Results

Input State	Senstive Transistor	Charge (fC)	LET (MeV*cm^2/mg)	Cross-Section (μm^2)	Effective Cross-Section (μm^2)	Output Ampli-tude (Volts)	Output Pulse-Width (ps)
A=1 B=0 C=0 D=0	T1 (NFET)	932	25.13	41.59	2.60	-2.74	150
		1310	35.33	41.59	2.60	-3.27	220
		1500	40.45	41.59	2.60	-3.3	250
		1570	42.34	41.59	2.60	-3.3	260
A=1 B=0 C=0 D=1	T4_1 (NFET)	393	10.60	31.76	1.99	-0.82	200
		623	16.80	31.76	1.99	-1.29	230
		733	19.77	31.76	1.99	-1.55	240
		775	20.90	31.76	1.99	-1.65	250
A=1 B=0 C=1 D=0	T1 (NFET)	796	21.47	41.59	2.60	-2.89	190
		1040	28.05	41.59	2.60	-3.3	240
		1170	31.55	41.59	2.60	-3.3	270
		1220	32.90	41.59	2.60	-3.3	290
A=1 B=0 C=1 D=1	T16 (NFET)	320	8.63	31.76	1.99	-0.79	310
		458	12.35	31.76	1.99	-1.19	330
		528	14.24	31.76	1.99	-1.4	330
		551	14.86	31.76	1.99	-1.44	350
A=1 B=1 C=0 D=0	T1 (NFET)	796	21.47	41.59	2.60	-2.89	190
		1040	28.05	41.59	2.60	-3.3	240
		1170	31.55	41.59	2.60	-3.3	270
		1220	32.90	41.59	2.60	-3.3	290
A=1 B=1 C=0 D=1	T4_1 (NFET)	389	10.49	31.76	1.99	-1.24	280
		598	16.13	31.76	1.99	-1.97	320
		695	18.74	31.76	1.99	-2.32	330
		725	19.55	31.76	1.99	-2.44	330
A=1 B=1 C=1 D=0	T5 (NFET)	648	17.48	41.59	2.60	-3.03	260
		771	20.79	41.59	2.60	-3.3	320
		834	22.49	41.59	2.60	-3.3	360
		860	23.19	41.59	2.60	-3.3	380
A=1 B=1 C=1 D=1	T1, T2, T3, T4 (PFET)	687	18.52	62.71	3.92	3.0	260
		795	21.44	62.71	3.92	3.3	320
		839	22.63	62.71	3.92	3.3	360
		858	23.14	62.71	3.92	3.3	380

Table 27. Nand4 SET Generation Results (Continued)

5. Nor2

The Nor2 (two input Nor gate) has four different input possibilities. Figure 61 shows the schematic, and Table 28 shows the cross-sections and LETs for each of the sensitive regions with the resulting amplitude and pulsewidth at the output.

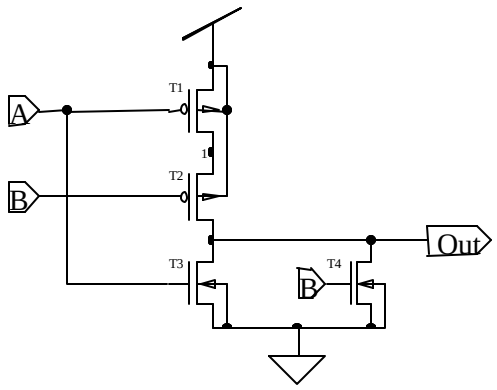


Figure 61. Nor2 Schematic (after [53])

Input State	Senstive Transistor	Charge (fC)	LET (MeV*cm²/mg)	Cross-Section (μm²)	Effective Cross-Section (μm²)	Output Ampli-tude (Volts)	Output Pulse-Width (ps)
A=0 B=0	T1,T2 (NFETS)	378	10.19	63.52	15.88	-3.0	290
		455	12.27	63.52	15.88	-3.3	360
		482	13.00	63.52	15.88	-3.3	390
		488	13.16	63.52	15.88	-3.3	400
A=0 B=1	T7 (PFET)	631	17.02	29.98	7.50	2.5	120
		947	25.54	29.98	7.50	3.25	230
		1030	27.78	29.98	7.50	3.3	260
		1080	29.13	29.98	7.50	3.3	270
A=1 B=0	T8 (PFET)	222	5.99	21.12	5.28	0.341	180
		350	9.44	21.12	5.28	0.467	230
		400	10.79	21.12	5.28	0.508	240
		414	11.17	21.12	5.28	0.522	250
A=1 B=1	T7 (PFET)	881	23.76	29.98	7.50	2.07	90
		1550	41.80	29.98	7.50	3.2	180
		1760	47.46	29.98	7.50	3.3	240
		1870	50.43	29.98	7.50	3.3	240

Table 28. Nor2 SET Generation Results

6. Nor3

The Nor3(three input Nor gate) has eight different input possibilities. Figure 62 shows the schematic, and the Table 29 shows the cross-sections and LETs for each of the sensitive regions with the resulting amplitude and pulsewidth at the output.

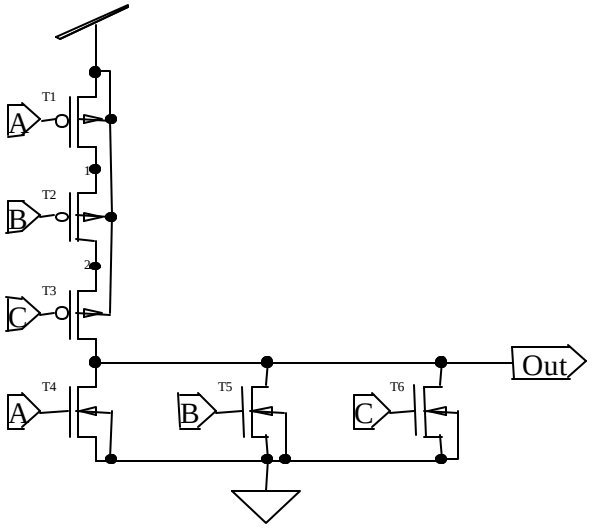


Figure 62. Nor3 Schematic (after [54])

Input State	Senstive Transistor	Charge (fC)	LET (MeV*cm^2/mg)	Cross-Section (μm^2)	Effective Cross-Section (μm^2)	Output Ampli-tude (Volts)	Output Pulse-Width (ps)
A=0 B=0 C=0	T2, T3_1, T4_1 (NFETS)	504	13.59	45.56	5.70	-2.79	480
		594	16.02	45.56	5.70	-3.27	570
		626	16.88	45.56	5.70	-3.3	620
		639	17.23	45.56	5.70	-3.3	650
A=0 B=0 C=1	T7 (PFET)	672	18.12	29.98	3.75	2.3	140
		993	26.78	29.98	3.75	3.24	210
		1150	31.01	29.98	3.75	3.3	250
		1220	32.90	29.98	3.75	3.3	280
A=0 B=1 C=0	T8 (PFET)	229	6.17	21.12	2.64	0.3	190
		353	9.52	21.12	2.64	0.43	240
		422	11.38	21.12	2.64	0.49	250
		451	12.16	21.12	2.64	0.51	260
A=0 B=1 C=1	T7 (PFET)	852	22.98	29.98	3.75	1.8	90
		1500	40.45	29.98	3.75	3.21	160
		1830	49.35	29.98	3.75	3.3	220
		1960	52.86	29.98	3.75	3.3	240
A=1 B=0 C=0	T6_1 (PFET)	211	5.69	21.12	2.64	0.13	280
		286	7.71	21.12	2.64	0.19	300
		320	8.63	21.12	2.64	0.22	300
		335	9.03	21.12	2.64	0.23	300
A=1 B=0 C=1	T7 (PFET)	852	22.98	21.12	2.64	1.8	90
		1500	40.45	21.12	2.64	3.21	160
		1830	49.35	21.12	2.64	3.3	220
		1960	52.86	21.12	2.64	3.3	240
A=1 B=1 C=0	T8 (PFET)	231	6.23	21.12	2.64	0.17	170
		357	9.63	21.12	2.64	0.22	220
		427	11.52	21.12	2.64	0.24	240
		454	12.24	21.12	2.64	0.24	250
A=1 B=1 C=1	T7 (PFET)	977	26.35	21.12	2.64	1.5	80
		1990	53.67	21.12	2.64	3.15	150
		2500	67.42	21.12	2.64	3.3	210
		2710	73.0	21.12	2.64	3.3	230

Table 29. Nor3 SET Generation Results

7. Nor4

The Nor4 (four input Nor gate) has sixteen different input possibilities. Figure 63 shows the schematic, and Tables 30 and 31 shows the cross-sections and LETs for each of the sensitive regions with the resulting amplitude and pulsewidth at the output.

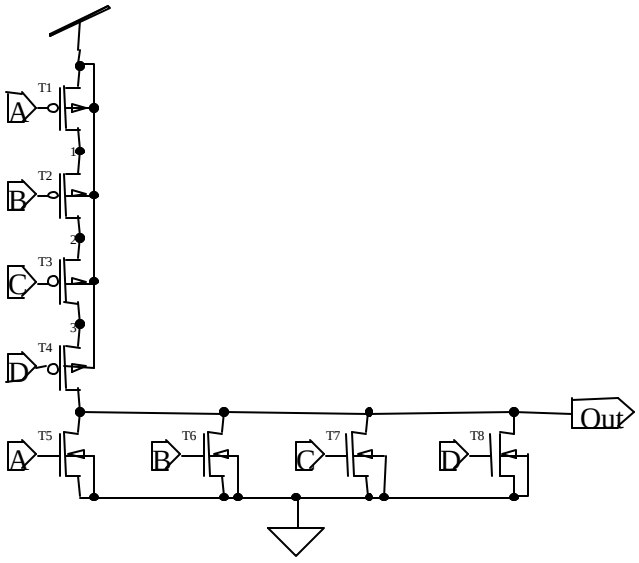


Figure 63. Nor4 Schematic (after [55])

Input State	Senstive Transistor	Charge (fC)	LET (MeV*cm^2/mg)	Cross-Section (μm^2)	Effective Cross-Section (μm^2)	Output Ampli-tude (Volts)	Output Pulse-Width (ps)
A=0 B=0 C=0 D=0	T5, T4_1, T3_1, T1 (NFETS)	660	15.83	127.04	7.94	-3.1	660
		760	17.29	127.04	7.94	-3.3	760
		810	17.93	127.04	7.94	-3.3	810
		830	18.20	127.04	7.94	-3.3	830
A=0 B=0 C=0 D=1	T2 (PFET)	392	10.57	29.98	1.87	0.61	180
		624	16.83	29.98	1.87	0.9	220
		747	20.15	29.98	1.87	1.04	230
		791	21.33	29.98	1.87	1.11	230
A=0 B=0 C=1 D=0	T8 (PFET)	343	9.25	21.12	1.32	0.272	210
		378	10.19	21.12	1.32	0.446	240
		432	11.65	21.12	1.32	0.489	260
		448	12.08	21.12	1.32	0.505	260
A=0 B=0 C=1 D=1	T2 (PFET)	1120	30.20	29.98	1.87	2.56	120
		1660	44.77	29.98	1.87	3.3	190
		1920	51.78	29.98	1.87	3.3	230
		2020	54.48	29.98	1.87	3.3	230
A=0 B=1 C=0 D=0	T9 (PFET)	236	6.36	21.12	1.32	0.155	290
		302	8.14	21.12	1.32	0.202	300
		339	9.14	21.12	1.32	0.220	310
		349	9.41	21.12	1.32	0.227	310
A=0 B=1 C=0 D=1	T2 (PFET)	1120	30.20	29.98	1.87	2.56	120
		1660	44.77	29.98	1.87	3.3	190
		1920	51.78	29.98	1.87	3.3	230
		2020	54.48	29.98	1.87	3.3	230
A=0 B=1 C=1 D=0	T8 (PFET)	276	7.44	21.12	1.32	0.187	180
		382	10.30	21.12	1.32	0.228	230
		436	11.76	21.12	1.32	0.24	250
		453	12.22	21.12	1.32	0.245	260
A=0 B=1 C=1 D=1	T2 (PFET)	1370	36.95	29.98	1.87	2.3	90
		2210	59.60	29.98	1.87	3.23	180
		2610	70.39	29.98	1.87	3.3	220
		2770	74.70	29.98	1.87	3.3	230

Table 30. Nor4 SET Generation Results

Input State	Senstive Transistor	Charge (fC)	LET (MeV*cm^2/mg)	Cross-Section (μm^2)	Effective Cross-Section (μm^2)	Output Ampli-tude (Volts)	Output Pulse-Width (ps)
A=1 B=0 C=0 D=0	T10_1 (PFET)	249	6.72	21.12	1.32	0.062	500
		284	7.66	21.12	1.32	0.107	460
		306	8.25	21.12	1.32	0.120	430
		314	8.47	21.12	1.32	0.124	430
A=1 B=0 C=0 D=1	T2 (PFET)	1120	30.20	29.98	1.87	2.56	120
		1660	44.77	29.98	1.87	3.3	190
		1920	51.78	29.98	1.87	3.3	230
		2020	54.48	29.98	1.87	3.3	230
A=1 B=0 C=1 D=0	T8 (PFET)	276	7.44	21.12	1.32	0.187	180
		382	10.30	21.12	1.32	0.228	230
		436	11.76	21.12	1.32	0.24	250
		453	12.22	21.12	1.32	0.245	260
A=1 B=0 C=1 D=1	T2 (PFET)	1370	36.95	29.98	1.87	2.3	90
		2210	59.60	29.98	1.87	3.23	180
		2610	70.39	29.98	1.87	3.3	220
		2770	74.70	29.98	1.87	3.3	230
A=1 B=1 C=0 D=0	T9 (PFET)	234	6.31	21.12	1.32	0.083	270
		293	7.90	21.12	1.32	0.105	280
		321	8.66	21.12	1.32	0.113	300
		328	8.85	21.12	1.32	0.115	310
A=1 B=1 C=0 D=1	T2 (PFET)	1370	36.95	29.98	1.87	2.3	90
		2210	59.60	29.98	1.87	3.23	180
		2610	70.39	29.98	1.87	3.3	220
		2770	74.70	29.98	1.87	3.3	230
A=1 B=1 C=1 D=0	T8 (PFET)	275	7.42	21.12	1.32	0.129	180
		390	10.52	21.12	1.32	0.152	230
		438	11.81	21.12	1.32	0.158	250
		455	12.27	21.12	1.32	0.16	250
A=1 B=1 C=1 D=1	T4 (PFET)	1570	42.34	29.98	1.87	2.08	80
		2750	74.16	29.98	1.87	3.2	150
		3310	89.27	29.98	1.87	3.3	210
		3520	94.93	29.98	1.87	3.3	230

Table 31. Nor4 SET Generation Results (Continued)

8. Xor2

The Xor2 (two-input XOR gate) has four different input possibilities. Figure 64 shows the schematic, and Table 32 shows the cross-sections and LETs for each of the sensitive regions with the resulting amplitude and pulsewidth at the output.

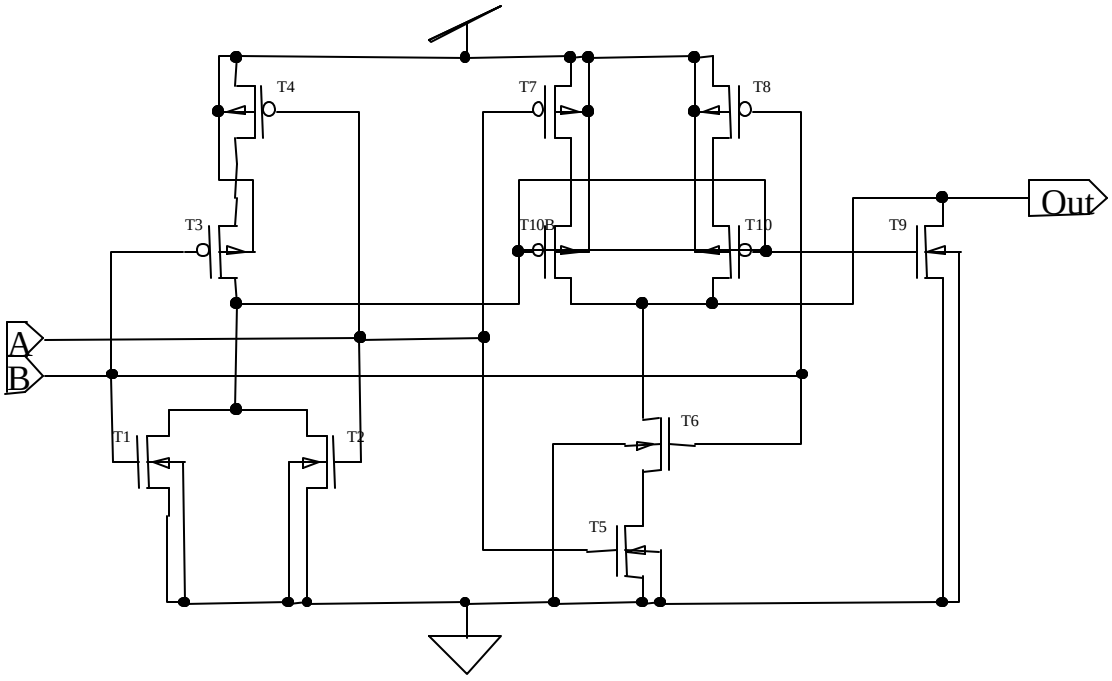


Figure 64. Xor2 Schematic (after [46])

Input State	Senstive Transistor	Charge (fC)	LET (MeV*cm^2/mg)	Cross-Section (μm^2)	Effective Cross-Section (μm^2)	Output Ampli-tude (Volts)	Output Pulse-Width (ps)
A=0 B=0	T1, T2 (NFETS)	346	9.33	55.27	13.82	2.48	330
		392	10.57	55.27	13.82	2.93	400
		407	10.98	55.27	13.82	3.02	430
		409	11.03	55.27	13.82	3.04	430
A=0 B=0	T10B, T10 (PFETS)	598	16.13	33.73	8.43	3.1	180
		755	20.36	33.73	8.43	3.3	260
		801	21.60	33.73	8.43	3.3	270
		809	21.82	33.73	8.43	3.3	270
A=0 B=1	T6_1, T9 (NFETS)	402	10.84	51.78	12.94	-3.15	340
		447	12.06	51.78	12.94	-3.3	390
		468	12.62	51.78	12.94	-3.3	420
		476	12.84	51.78	12.94	-3.3	420
A=1 B=0	T3 (PFET)	619	16.69	24.50	6.13	-1.84	360
		790	21.31	24.50	6.13	-3.08	440
		870	23.46	24.50	6.13	-3.25	470
		893	24.22	24.50	6.13	-3.3	480
A=1 B=0	T9 (NFET)	463	12.49	25.06	6.26	-3.09	410
		519	14.00	25.06	6.26	-3.3	470
		543	14.64	25.06	6.26	-3.3	510
		552	14.89	25.06	6.26	-3.3	530
A=1 B=0	T5 (PFET)	318	8.58	20.43	5.11	-1.84	410
		436	11.76	20.43	5.11	-2.65	440
		473	12.76	20.43	5.11	-2.91	450
		483	13.03	20.43	5.11	-2.98	450
A=1 B=1	T6 (PFET)	215	5.80	21.58	5.39	0.588	210
		285	7.69	21.58	5.39	0.759	240
		316	8.52	21.58	5.39	0.835	250
		324	8.74	21.58	5.39	0.866	250
A=1 B=1	T7 (PFET)	215	5.80	21.58	5.39	0.588	210
		285	7.69	21.58	5.39	0.759	240
		316	8.52	21.58	5.39	0.835	250
		324	8.74	21.58	5.39	0.866	250

Table 32. Xor2 SET Generation Results

9. Mux2

The Mux2 (2-to-1 Multiplexer) has three inputs: A, B, and Sel. Thus, there are eight different input possibilities. Figure 65 shows the schematic, and Tables 33 and 34 show the cross-sections and LETs for each of the sensitive regions with the resulting amplitude and pulsewidth at the output.

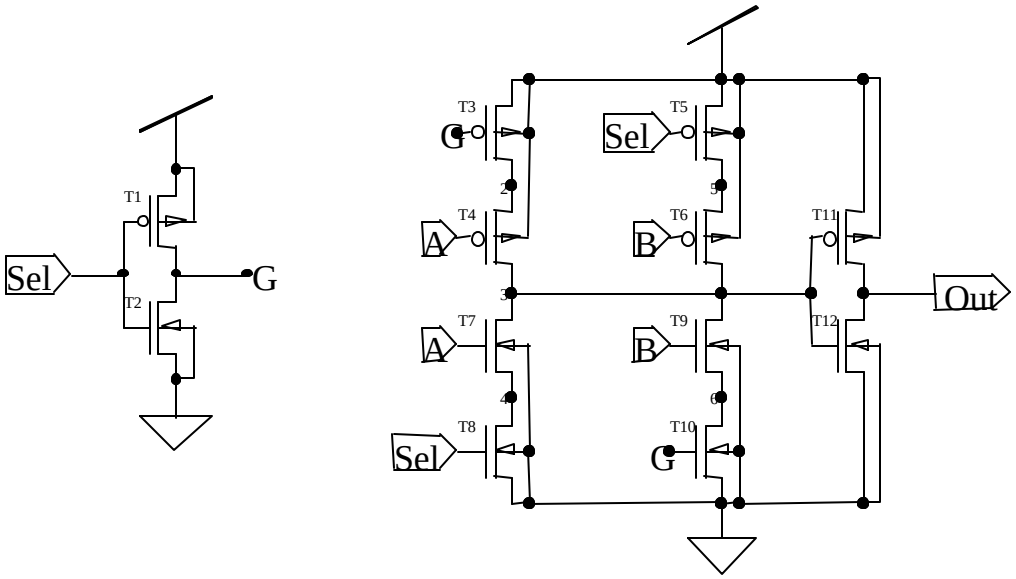


Figure 65. Mux2 Schematic (after [56])

Input State	Senstive Transistor	Charge (fC)	LET (MeV*cm^2/mg)	Cross-Section (μm^2)	Effective Cross-Section (μm^2)	Output Ampli-tude (Volts)	Output Pulse-Width (ps)
A=0 B=0 Sel=0	T14, T12 (NFETS)	452	12.19	196.34	24.54	2.44	360
		501	13.51	196.34	24.54	2.92	450
		522	14.08	196.34	24.54	3.05	470
		530	14.29	196.34	24.54	3.08	480
A=0 B=0 Sel=0	T1 (PFET)	718	19.36	29.98	3.75	2.95	150
		962	25.94	29.98	3.75	3.27	220
		1070	28.86	29.98	3.75	3.3	240
		1120	30.20	29.98	3.75	3.3	260
A=0 B=0 Sel=1	T14, T12 (NFETS)	452	12.19	196.34	24.54	2.44	360
		501	13.51	196.34	24.54	2.92	450
		522	14.08	196.34	24.54	3.05	470
		530	14.29	196.34	24.54	3.08	480
A=0 B=1 Sel=0	T6_1 (PFET)	559	15.08	16.15	2.02	-3.24	350
		676	18.23	16.15	2.02	-3.3	440
		727	19.61	16.15	2.02	-3.3	480
		747	20.15	16.15	2.02	-3.3	500
A=0 B=1 Sel=0	T2 (NFET)	507	13.67	41.59	5.20	-3.1	220
		619	16.69	41.59	5.20	-3.3	270
		676	18.23	41.59	5.20	-3.3	300
		696	18.77	41.59	5.20	-3.3	310
A=0 B=1 Sel=1	T14 (NFET)	462	12.49	24.54	3.07	2.61	420
		523	14.10	24.54	3.07	3.03	500
		542	14.62	24.54	3.07	3.11	530
		545	14.70	24.54	3.07	3.14	550
A=0 B=1 Sel=1	T1 (PFET)	719	19.39	29.98	3.75	2.93	140
		780	21.04	29.98	3.75	3.28	210
		1080	29.13	29.98	3.75	3.3	250
		1120	30.20	29.98	3.75	3.3	260
A=0 B=1 Sel=1	T5_1 (PFET)	552	14.89	29.98	3.75	0.067	140
		707	19.07	29.98	3.75	0.68	210
		780	21.04	29.98	3.75	1.45	250
		807	21.76	29.98	3.75	1.7	260

Table 33. Mux2 SET Generation Results

Input State	Senstive Transistor	Charge (fC)	LET (MeV*cm^2/mg)	Cross-Section (μm^2)	Effective Cross-Section (μm^2)	Output Ampli-tude (Volts)	Output Pulse-Width (ps)
A=1 B=0 Sel=0	T12 (NFET)	464	12.51	24.54	3.07	2.63	410
		514	13.86	24.54	3.07	3.04	500
		536	14.46	24.54	3.07	3.11	540
		547	14.75	24.54	3.07	3.16	540
A=1 B=0 Sel=0	T10_1 (NFET)	294	7.93	18.07	2.26	0.151	180
		407	10.98	18.07	2.26	1.62	260
		447	12.06	18.07	2.26	1.84	310
		462	12.46	18.07	2.26	2.03	340
A=1 B=0 Sel=0	T1 (PFET)	179	19.39	29.98	3.75	2.93	150
		963	25.97	29.98	3.75	3.28	210
		1080	29.13	29.98	3.75	3.3	250
		1130	30.47	29.98	3.75	3.3	260
A=1 B=0 Sel=1	T12 (NFET)	559	15.08	24.54	3.07	-3.26	360
		674	18.18	24.54	3.07	-3.3	440
		728	19.63	24.54	3.07	-3.3	480
		745	20.09	24.54	3.07	-3.3	490
A=1 B=0 Sel=1	T2 (NFET)	507	13.67	41.59	5.20	-3.1	220
		619	16.69	41.59	5.20	-3.3	270
		676	18.23	41.59	5.20	-3.3	300
		696	18.77	41.59	5.20	-3.3	310
A=1 B=1 Sel=0	T7, T6_1 (PFET)	579	15.61	32.31	4.04	-3.23	360
		686	18.50	32.31	4.04	-3.3	430
		735	19.82	32.31	4.04	-3.3	470
		752	20.28	32.31	4.04	-3.3	490
A=1 B=1 Sel=0	T2 (NFET)	506	16.65	41.59	5.20	-3.11	210
		619	16.69	41.59	5.20	-3.3	270
		674	18.18	41.59	5.20	-3.3	310
		696	18.77	41.59	5.20	-3.3	320
A=1 B=1 Sel=1	T7, T6_1 (PFET)	579	15.61	32.31	4.04	-3.23	360
		686	18.50	32.31	4.04	-3.3	430
		735	19.82	32.31	4.04	-3.3	470
		752	20.28	32.31	4.04	-3.3	490
A=1 B=1 Sel=1	T2 (NFET)	506	13.65	41.59	5.20	-3.11	210
		619	16.69	41.59	5.20	-3.3	270
		674	18.18	41.59	5.20	-3.3	310
		696	18.77	41.59	5.20	-3.3	320

Table 34. Mux2 SET Generation Results (Continued)

10. Buf4

The Buf4 (High-Drive Buffer) has two input possibilities. Figure 66 shows the schematic, and Table 35 shows the cross-sections and LETs for each of the sensitive regions with the resulting amplitude and pulsewidth at the output.

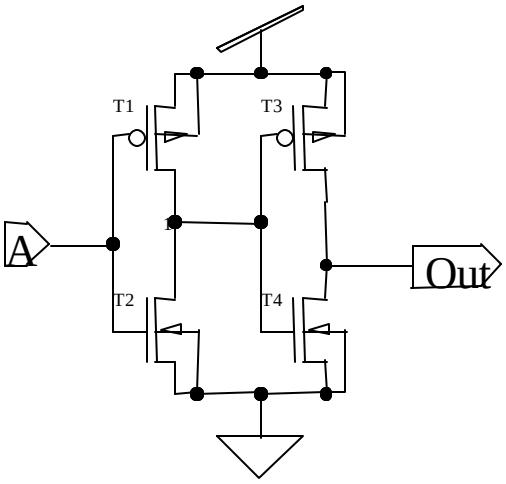


Figure 66. Buf4 Schematic (after [57])

Input State	Senstive Transistor	Charge (fC)	LET (MeV*cm²/mg)	Cross-Section (μm²)	Effective Cross-Section (μm²)	Output Amplitude (Volts)	Output Pulse-Width (ps)
A=0	T1 (NFET)	713	19.23	30.05	7.51	3	340
		837	22.57	30.05	7.51	3.3	470
		888	23.95	30.05	7.51	3.3	520
		906	24.43	30.05	7.51	3.3	540
A=0	T10 (PFET)	1650	44.5	168.94	42.24	1.91	90
		3040	82.0	168.94	42.24	3.2	160
		3750	101.1	168.94	42.24	3.3	220
		4010	108.1	168.94	42.24	3.3	230
A=1	T2 (NFET)	1180	31.82	31.76	7.94	-2.45	140
		1800	48.5	31.76	7.94	-3.24	210
		2100	56.6	31.76	7.94	-3.3	240
		2220	59.9	31.76	7.94	-3.3	260
A=1	T9 (PFET)	858	23.14	21.67	5.42	-3.3	310
		1050	28.32	21.67	5.42	-3.3	420
		1130	30.74	21.67	5.42	-3.3	470
		1170	31.55	21.67	5.42	-3.3	480

Table 35. Buf4 SET Generation Results

C. ANALOG PROPAGATION

For each of the standard cells, the relationship between the input and output SET amplitude and pulsewidth was determined. These results are summarized in Tables 36 and 37.

Standard Cell	Transition	Input Amplitude (Volts)	Input Pulsewidth (Picoseconds)	Output Amplitude (Volts)	Output Pulsewidth (Picoseconds)
NAND2	0-1-0	2.6	110	-1.37	160
		3.11	150	-2.35	220
		3.25	190	-2.87	250
		3.3	240	-3.2	300
NAND2	1-0-1	-2.92	180	0.89	160
		-3.19	210	1.49	200
		-3.3	240	2.0	220
		-3.3	290	2.63	260
NAND3	0-1-0	1.54	150	-0.21	210
		2.47	220	-1.43	210
		3.0	2.8	-2.26	280
		3.3	440	-3.05	450
NAND3	1-0-1	-2.39	180	0.38	190
		-3.3	240	1.41	260
		-3.3	360	2.34	350
		-3.3	490	3.09	440
NAND4	0-1-0	1.5	160	-0.06	470
		2.48	220	-0.54	350
		2.77	250	-0.72	330
		2.84	250	-0.79	340
NAND4	1-0-1	-2.65	220	0.43	330
		-3.25	260	0.88	380
		-3.3	290	1.08	410
		-3.3	340	1.28	440
NOR2	0-1-0	2.64	120	-1.54	270
		3.14	170	-3.11	330
		3.3	220	-3.3	410
		3.3	250	-3.3	450
NOR2	1-0-1	-2.93	190	0.49	130
		-3.22	230	1.03	150
		-3.28	260	1.43	160
		-3.3	310	1.7	190

Table 36. SET Analog Propagation Results

Standard Cell	Transition	Input Amplitude (Volts)	Input Pulsewidth (Picoseconds)	Output Amplitude (Volts)	Output Pulsewidth (Picoseconds)
NOR3	0-1-0	1.5	170	-0.30	480
		2.11	200	-0.96	440
		2.68	230	-1.90	610
		2.85	250	-2.36	560
NOR3	1-0-1	-3.02	240	0.13	250
		-3.3	280	0.16	240
		-3.3	320	0.27	230
		-3.3	330	0.28	220
NOR4	0-1-0	1.5	160	-0.33	540
		2.1	200	-0.93	550
		2.7	230	-1.8	630
		2.9	240	-2.2	700
NOR4	1-0-1	-2.64	220	0.03	760
		-3.26	260	0.07	450
		-3.3	290	0.09	400
		-3.3	340	0.11	370
XOR2	0-1-0	1.69	170	-0.4	280
		2.34	200	-1.02	320
		2.87	230	-1.84	370
		3.0	250	-2.23	390
XOR2	1-0-1	-2.89	200	0.52	170
		-3.3	250	1.0	190
		-3.3	300	1.4	210
		-3.3	320	1.5	230
MUX2	0-1-0	3.0	250	0.26	200
		3.2	300	1.3	270
		3.3	410	2.9	530
		3.3	420	3.0	490
MUX2	1-0-1	-3.3	320	-0.07	260
		-3.3	390	-0.3	210
		-3.3	490	-1.15	240
		-3.3	690	-3.3	450
BUF4	0-1-0	1.73	160	0.003	260
		2.73	220	0.045	150
		3.0	240	0.13	160
		3.1	260	0.18	170
BUF4	1-0-1	-2.96	210	-0.02	210
		-3.3	280	-0.13	180
		-3.3	310	-0.24	180
		-3.3	320	-0.29	180

Table 37. SET Analog Propagation Results (Continued)

D. EFFECTIVE CROSS-SECTIONS OF DATAPATHS

The purpose of this section is to show the effective cross-sections of the datapaths that were not shown in Chapter IV (the AND instruction was shown in Chapter IV). Tables 38 through 53 show the effective cross-section for the execute datapath for these remaining instructions.

Logic Block	**cross-section (μm^2)**	**P_{scl}**	**P_{ap}**	**$P_{latched}$ (1/MHz)**	**d1 (1/MHz)**	**s*d1 (μm^2/Mhz)**
Mux2_1	131.6169	0.5	1	1.50E-04	7.50E-05	9.87E-03
Mux2_2	131.6169	0.5	1	1.50E-04	7.50E-05	9.87E-03
Mux2_3	131.6169	0.5	1	1.50E-04	7.50E-05	9.87E-03
Mux2_4	131.6169	0.5	1	1.50E-04	7.50E-05	9.87E-03
Mux2_5	131.6169	0.5	1	1.50E-04	7.50E-05	9.87E-03
Xor2_1	63.48	1	1	1.50E-04	1.50E-04	9.52E-03
Nand2_1	41.7522	0.25	1	1.50E-04	3.75E-05	1.57E-03
Nand2_2	41.7522	0.25	1	1.50E-04	3.75E-05	1.57E-03
Nand2_3	41.7522	0.25	1	1.50E-04	3.75E-05	1.57E-03
Nand3_1	40.54	1	1	1.50E-04	1.50E-04	6.08E-03
Xor2_2	63.48	1	1	1.50E-04	1.50E-04	9.52E-03
Mux2_6	131.6169	1	1	1.50E-04	1.50E-04	1.97E-02
Mux2_7	131.6169	1	1	1.50E-04	1.50E-04	1.97E-02
Mux2_8	131.6169	1	1	1.50E-04	1.50E-04	1.97E-02
Mux2_9	131.6169	1	1	1.50E-04	1.50E-04	1.97E-02
					Total Effective Cross-Section	1.58E-01

Table 38. Effective Cross-Section for AND and ANDUI Logic Datapaths

Logic Block	cross-section (mm2)	P_{scl}	P_{ap}	$P_{latched}$ (1/MHz)	d1 (1/MHz)	s*d1 (mm²/Mhz)
Mux2_1	1.32E+02	0.5	1	1.50E-04	7.50E-05	9.87E-03
Mux2_2	1.32E+02	0.5	1	1.50E-04	7.50E-05	9.87E-03
Mux2_3	1.32E+02	0.5	1	1.50E-04	7.50E-05	9.87E-03
Mux2_4	1.32E+02	0.5	1	1.50E-04	7.50E-05	9.87E-03
Xor2_1	6.35E+01	1	1	1.50E-04	1.50E-04	9.52E-03
Nand2_1	4.18E+01	0.25	1	1.50E-04	3.75E-05	1.57E-03
Nand2_2	4.18E+01	0.25	1	1.50E-04	3.75E-05	1.57E-03
Nand2_3	4.18E+01	0.25	1	1.50E-04	3.75E-05	1.57E-03
Nand3_1	4.05E+01	1	1	1.50E-04	1.50E-04	6.08E-03
Xor2_2	6.35E+01	1	1	1.50E-04	1.50E-04	9.52E-03
Mux2_5	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
Mux2_6	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
Mux2_7	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
Mux2_8	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
					Total Effective Cross-Section	1.48E-01

Table 39. ADD and ADDUI Logic Datapath Effective Cross-section

Logic Block	Cross-Section (mm^2)	P_{scl}	P_{ap}	$P_{latched}$ (1/MHz)	d1 (1/MHz)	s*d1 (mm^2/Mhz)
Mux2_1	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
mux2_2	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
mux2_3	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
mux2_4	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
mux2_5	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
mux2_6	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
					Total Effective Cross-Section	1.18E-01

Table 40. LHI Logic Datapath Effective Cross-Section

Logic Block	Cross-Section (μm^2)	P_{scl}	P_{ap}	$P_{latched}$ (1/MHz)	d1 (1/MHz)	s*d1 (μm^2/Mhz)
Mux2_1	1.32E+02	0.5	1	1.50E-04	7.50E-05	9.87E-03
Mux2_2	1.32E+02	0.5	1	1.50E-04	7.50E-05	9.87E-03
Mux2_3	1.32E+02	0.5	1	1.50E-04	7.50E-05	9.87E-03
Mux2_4	1.32E+02	0.5	1	1.50E-04	7.50E-05	9.87E-03
Mux2_5	1.32E+02	0.5	1	1.50E-04	7.50E-05	9.87E-03
Inv_1	3.58E+01	1	1	1.50E-04	1.50E-04	5.37E-03
Xor2_1	6.35E+01	1	1	1.50E-04	1.50E-04	9.52E-03
Nand2_1	4.18E+01	0.25	1	1.50E-04	3.75E-05	1.57E-03
Nand2_2	4.18E+01	0.25	1	1.50E-04	3.75E-05	1.57E-03
Nand2_3	4.18E+01	0.25	1	1.50E-04	3.75E-05	1.57E-03
Nand3_1	4.05E+01	1	1	1.50E-04	1.50E-04	6.08E-03
Xor2_2	6.35E+01	1	1	1.50E-04	1.50E-04	9.52E-03
Mux2_6	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
Mux2_7	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
Mux2_8	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
Mux2_9	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
					Total Effective Cross-Section	1.64E-01

Table 41. SUB and SUBUI Logic Datapath Effective Cross-Section

Logic Block	Cross-Section (μm^2)	P_{scl}	P_{ap}	$P_{latched}$ (1/MHz)	d1 (1/MHz)	s*d1 (μm^2/Mhz)
Mux2_1	1.32E+02	0.5	1	1.50E-04	7.50E-05	9.87E-03
mux2_2	1.32E+02	0.5	1	1.50E-04	7.50E-05	9.87E-03
mux2_3	1.32E+02	0.5	1	1.50E-04	7.50E-05	9.87E-03
mux2_4	1.32E+02	0.5	1	1.50E-04	7.50E-05	9.87E-03
Xor2_1	6.35E+01	1	1	1.50E-04	1.50E-04	9.52E-03
mux2_5	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
mux2_6	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
mux2_7	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
mux2_8	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
mux2_9	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
					Total Effective Cross-Section	1.48E-01

Table 42. XOR and XORI Logic Datatpath Effective Cross-Section

Logic Block	Cross-Section (μm^2)	P_{scl}	P_{ap}	$P_{latched}$ (1/MHz)	d1 (1/MHz)	s*d1 (μm^2/Mhz)
Mux2_1	1.32E+02	0.5	1	1.50E-04	7.50E-05	9.87E-03
Mux2_2	1.32E+02	0.5	1	1.50E-04	7.50E-05	9.87E-03
Mux2_3	1.32E+02	0.5	1	1.50E-04	7.50E-05	9.87E-03
Mux2_4	1.32E+02	0.5	1	1.50E-04	7.50E-05	9.87E-03
Inv_1	3.58E+01	1	1	1.50E-04	1.50E-04	5.37E-03
Xor2_1	6.35E+01	1	1	1.50E-04	1.50E-04	9.52E-03
Nand2_1	4.18E+01	0.25	1	1.50E-04	3.75E-05	1.57E-03
Nand2_2	4.18E+01	0.25	1	1.50E-04	3.75E-05	1.57E-03
Nand2_3	4.18E+01	0.25	1	1.50E-04	3.75E-05	1.57E-03
Nand3_1	4.05E+01	1	1	1.50E-04	1.50E-04	6.08E-03
Xor2_2	6.35E+01	1	1	1.50E-04	1.50E-04	9.52E-03
Mux2_5	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
Mux2_6	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
Mux2_7	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
Mux2_8	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
					Total Effective Cross-Section	1.54E-01

Table 43. SUBI Logic Datapath Effective Cross-Section

Logic Block	Cross-Section (μm^2)	P_{scl}	P_{ap}	$P_{latched}$ (1/MHz)	d1 (1/MHz)	s*d1 (μm^2/Mhz)
Mux2_1	1.32E+02	0.5	1	1.50E-04	7.50E-05	9.87E-03
mux2_2	1.32E+02	0.5	1	1.50E-04	7.50E-05	9.87E-03
mux2_3	1.32E+02	0.5	1	1.50E-04	7.50E-05	9.87E-03
mux2_4	1.32E+02	0.5	1	1.50E-04	7.50E-05	9.87E-03
Nor2_1	3.62E+01	1	1	1.50E-04	1.50E-04	5.42E-03
Inv_1	3.58E+01	1	1	1.50E-04	1.50E-04	5.37E-03
mux2_6	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
mux2_7	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
mux2_8	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
mux2_9	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
mux2_10	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
					Total Effective Cross-Section	1.49E-01

Table 44. OR and ORI Logic Datapath Effective Cross-Section

Logic Block	Cross-Section (μm^2)	P_{scl}	P_{ap}	$P_{latched}$ (1/MHz)	d1 (1/MHz)	s*d1 (μm^2/Mhz)
Mux2_1	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
mux2_2	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
mux2_3	1.32E+02	0.15625	1	1.50E-04	2.34E-05	3.08E-03
mux2_4	1.32E+02	0.15625	1	1.50E-04	2.34E-05	3.08E-03
mux2_5	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
mux2_6	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
mux2_7	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
mux2_8	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
mux2_9	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
mux2_10	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
mux2_11	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
mux2_12	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
mux2_13	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
mux2_14	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
mux2_15	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
mux2_16	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
					Total Effective Cross-Section	2.83E-01

Table 45. SLL, SLLI, SRL, SRLI, SRA, SRAI Logic Datapath Effective Cross-Section

Logic Block	Cross-Section (μm^2)	P_{scl}	P_{ap}	$P_{latched}$ (1/MHz)	d1 (1/MHz)	s*d1 (μm^2/Mhz)
Mux2_1	1.32E+02	0.000488	1	1.50E-04	7.32E-08	9.64E-06
Mux2_2	1.32E+02	0.000488	1	1.50E-04	7.32E-08	9.64E-06
Mux2_3	1.32E+02	0.000488	1	1.50E-04	7.32E-08	9.64E-06
Mux2_4	1.32E+02	0.000488	1	1.50E-04	7.32E-08	9.64E-06
Mux2_5	1.32E+02	0.000488	1	1.50E-04	7.32E-08	9.64E-06
Inv_1	3.58E+01	0.000977	1	1.50E-04	1.46E-07	5.24E-06
Xor2_1	6.35E+01	0.000977	1	1.50E-04	1.46E-07	9.30E-06
Nand2_1	4.18E+01	0.000244	1	1.50E-04	3.66E-08	1.53E-06
Nand2_2	4.18E+01	0.000244	1	1.50E-04	3.66E-08	1.53E-06
Nand2_3	4.18E+01	0.000244	1	1.50E-04	3.66E-08	1.53E-06
Nand3_1	4.05E+01	0.000977	1	1.50E-04	1.46E-07	5.94E-06
Xor2_2	6.35E+01	0.000977	1	1.50E-04	1.46E-07	9.30E-06
Nor4_1	3.61E+01	0.007813	1	1.50E-04	1.17E-06	4.23E-05
Nor4_2	3.61E+01	0.0625	1	1.50E-04	9.38E-06	3.38E-04
Mux2_6	1.32E+02	0.0625	1	1.50E-04	9.38E-06	1.23E-03
Mux2_7	1.32E+02	0.0625	1	1.50E-04	9.38E-06	1.23E-03
Mux2_8	1.32E+02	0.0625	1	1.50E-04	9.38E-06	1.23E-03
Mux2_9	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
Mux2_10	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
Mux2_11	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
Mux2_12	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
					Total Effective Cross-Section	8.31E-02

Table 46. SEQ and SEQI Logic Datapath Effective Cross-Section

Logic Block	Cross-Section (μm^2)	P_{scl}	P_{ap}	$P_{latched}$ (1/MHz)	d1 (1/MHz)	s*d1 (μm^2/Mhz)
Mux2_1	1.32E+02	0.000488	1	1.50E-04	7.32E-08	9.64E-06
Mux2_2	1.32E+02	0.000488	1	1.50E-04	7.32E-08	9.64E-06
Mux2_3	1.32E+02	0.000488	1	1.50E-04	7.32E-08	9.64E-06
Mux2_4	1.32E+02	0.000488	1	1.50E-04	7.32E-08	9.64E-06
Mux2_5	1.32E+02	0.000488	1	1.50E-04	7.32E-08	9.64E-06
Inv_1	3.58E+01	0.000977	1	1.50E-04	1.46E-07	5.24E-06
Xor2_1	6.35E+01	0.000977	1	1.50E-04	1.46E-07	9.30E-06
Nand2_1	4.18E+01	0.000244	1	1.50E-04	3.66E-08	1.53E-06
Nand2_2	4.18E+01	0.000244	1	1.50E-04	3.66E-08	1.53E-06
Nand2_3	4.18E+01	0.000244	1	1.50E-04	3.66E-08	1.53E-06
Nand3_1	4.05E+01	0.000977	1	1.50E-04	1.46E-07	5.94E-06
Xor2_2	6.35E+01	0.000977	1	1.50E-04	1.46E-07	9.30E-06
Nor4_1	3.61E+01	0.007813	1	1.50E-04	1.17E-06	4.23E-05
Nor4_2	3.61E+01	0.0625	1	1.50E-04	9.38E-06	3.38E-04
Mux2_6	1.32E+02	0.0625	1	1.50E-04	9.38E-06	1.23E-03
Mux2_7	1.32E+02	0.0625	1	1.50E-04	9.38E-06	1.23E-03
Mux2_8	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
Mux2_9	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
Mux2_10	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
Mux2_11	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
					Total Effective Cross-Section	8.19E-02

Table 47. SNE and SNEI Logic Datapath Effective Cross-Section

Logic Block	Cross-Section (μm^2)	P_{scl}	P_{ap}	$P_{latched}$ (1/MHz)	d1 (1/MHz)	s*d1 (μm^2/Mhz)
Mux2_1	1.32E+02	0.000488	1	1.50E-04	7.32E-08	9.64E-06
Mux2_2	1.32E+02	0.000488	1	1.50E-04	7.32E-08	9.64E-06
Mux2_3	1.32E+02	0.000488	1	1.50E-04	7.32E-08	9.64E-06
Mux2_4	1.32E+02	0.000488	1	1.50E-04	7.32E-08	9.64E-06
Mux2_5	1.32E+02	0.000488	1	1.50E-04	7.32E-08	9.64E-06
Inv_1	3.58E+01	0.000977	1	1.50E-04	1.46E-07	5.24E-06
Xor2_1	6.35E+01	0.000977	1	1.50E-04	1.46E-07	9.30E-06
Nand2_1	4.18E+01	0.000244	1	1.50E-04	3.66E-08	1.53E-06
Nand2_2	4.18E+01	0.000244	1	1.50E-04	3.66E-08	1.53E-06
Nand2_3	4.18E+01	0.000244	1	1.50E-04	3.66E-08	1.53E-06
Nand3_1	4.05E+01	0.000977	1	1.50E-04	1.46E-07	5.94E-06
Xor2_2	6.35E+01	0.000977	1	1.50E-04	1.46E-07	9.30E-06
Mux2_6	1.32E+02	0.0625	1	1.50E-04	9.38E-06	1.23E-03
Mux2_7	1.32E+02	0.0625	1	1.50E-04	9.38E-06	1.23E-03
Mux2_8	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
Mux2_9	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
Mux2_10	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
Mux2_11	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
					Total Effective Cross-Section	8.15E-02

Table 48. SLT and SLTI Logic Datapath Effective Cross-Section

Logic Block	Cross-Section (μm^2)	P_{scl}	P_{ap}	$P_{latched}$ (1/MHz)	d1 (1/MHz)	s*d1 (μm^2/Mhz)
Mux2_1	1.32E+02	0.000488	1	1.50E-04	7.32E-08	9.64E-06
Mux2_2	1.32E+02	0.000488	1	1.50E-04	7.32E-08	9.64E-06
Mux2_3	1.32E+02	0.000488	1	1.50E-04	7.32E-08	9.64E-06
Mux2_4	1.32E+02	0.000488	1	1.50E-04	7.32E-08	9.64E-06
Mux2_5	1.32E+02	0.000488	1	1.50E-04	7.32E-08	9.64E-06
Inv_1	3.58E+01	0.000977	1	1.50E-04	1.46E-07	5.24E-06
Xor2_1	6.35E+01	0.000977	1	1.50E-04	1.46E-07	9.30E-06
Nand2_1	4.18E+01	0.000244	1	1.50E-04	3.66E-08	1.53E-06
Nand2_2	4.18E+01	0.000244	1	1.50E-04	3.66E-08	1.53E-06
Nand2_3	4.18E+01	0.000244	1	1.50E-04	3.66E-08	1.53E-06
Nand3_1	4.05E+01	0.000977	1	1.50E-04	1.46E-07	5.94E-06
Xor2_2	6.35E+01	0.000977	1	1.50E-04	1.46E-07	9.30E-06
Inv_1	3.58E+01	0.0625	1	1.50E-04	9.38E-06	3.36E-04
Mux2_6	1.32E+02	0.0625	1	1.50E-04	9.38E-06	1.23E-03
Mux2_7	1.32E+02	0.0625	1	1.50E-04	9.38E-06	1.23E-03
Mux2_7	1.32E+02	0.0625	1	1.50E-04	9.38E-06	1.23E-03
Mux2_8	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
Mux2_9	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
Mux2_10	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
Mux2_11	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
					Total Effective Cross-Section	8.31E-02

Table 49. SGE and SGEI Logic Datapath Effective Cross-Section

Logic Block	Cross-Section (μm^2)	P_{scl}	P_{ap}	$P_{latched}$ (1/MHz)	d1 (1/MHz)	s*d1 (μm^2/Mhz)
Mux2_1	1.32E+02	0.000488	1	1.50E-04	7.32E-08	9.64E-06
Mux2_2	1.32E+02	0.000488	1	1.50E-04	7.32E-08	9.64E-06
Mux2_3	1.32E+02	0.000488	1	1.50E-04	7.32E-08	9.64E-06
Mux2_4	1.32E+02	0.000488	1	1.50E-04	7.32E-08	9.64E-06
Mux2_5	1.32E+02	0.000488	1	1.50E-04	7.32E-08	9.64E-06
Inv_1	3.58E+01	0.000977	1	1.50E-04	1.46E-07	5.24E-06
Xor2_1	6.35E+01	0.000977	1	1.50E-04	1.46E-07	9.30E-06
Nand2_1	4.18E+01	0.000244	1	1.50E-04	3.66E-08	1.53E-06
Nand2_2	4.18E+01	0.000244	1	1.50E-04	3.66E-08	1.53E-06
Nand2_3	4.18E+01	0.000244	1	1.50E-04	3.66E-08	1.53E-06
Nand3_1	4.05E+01	0.000977	1	1.50E-04	1.46E-07	5.94E-06
Xor2_2	6.35E+01	0.000977	1	1.50E-04	1.46E-07	9.30E-06
Nor2_1	3.62E+01	0.0625	1	1.50E-04	9.38E-06	3.39E-04
Inv_2	3.58E+01	0.0625	1	1.50E-04	9.38E-06	3.36E-04
Mux2_6	1.32E+02	0.0625	1	1.50E-04	9.38E-06	1.23E-03
Mux2_7	1.32E+02	0.0625	1	1.50E-04	9.38E-06	1.23E-03
Mux2_7	1.32E+02	0.0625	1	1.50E-04	9.38E-06	1.23E-03
Mux2_8	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
Mux2_9	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
Mux2_10	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
Mux2_11	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
					Total Effective Cross-Section	8.34E-02

Table 50. SLE and SLEI Logic Datapath Effective Cross-Section

Logic Block	Cross-Section (μm^2)	P_{scl}	P_{ap}	$P_{latched}$ (1/MHz)	d1 (1/MHz)	s*d1 (μm^2/Mhz)
Mux2_1	1.32E+02	0.000488	1	1.50E-04	7.32E-08	9.64E-06
Mux2_2	1.32E+02	0.000488	1	1.50E-04	7.32E-08	9.64E-06
Mux2_3	1.32E+02	0.000488	1	1.50E-04	7.32E-08	9.64E-06
Mux2_4	1.32E+02	0.000488	1	1.50E-04	7.32E-08	9.64E-06
Mux2_5	1.32E+02	0.000488	1	1.50E-04	7.32E-08	9.64E-06
Inv_1	3.58E+01	0.000977	1	1.50E-04	1.46E-07	5.24E-06
Xor2_1	6.35E+01	0.000977	1	1.50E-04	1.46E-07	9.30E-06
Nand2_1	4.18E+01	0.000244	1	1.50E-04	3.66E-08	1.53E-06
Nand2_2	4.18E+01	0.000244	1	1.50E-04	3.66E-08	1.53E-06
Nand2_3	4.18E+01	0.000244	1	1.50E-04	3.66E-08	1.53E-06
Nand3_1	4.05E+01	0.000977	1	1.50E-04	1.46E-07	5.94E-06
Xor2_2	6.35E+01	0.000977	1	1.50E-04	1.46E-07	9.30E-06
Nor2_1	3.62E+01	0.0625	1	1.50E-04	9.38E-06	3.39E-04
Mux2_6	1.32E+02	0.0625	1	1.50E-04	9.38E-06	1.23E-03
Mux2_7	1.32E+02	0.0625	1	1.50E-04	9.38E-06	1.23E-03
Mux2_7	1.32E+02	0.0625	1	1.50E-04	9.38E-06	1.23E-03
Mux2_8	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
Mux2_9	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
Mux2_10	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
Mux2_11	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
					Total Effective Cross-Section	8.31E-02

Table 51. SGT and SGTI Logic Datatpath Effective Cross-Section

Logic Block	Cross-Section (μm^2)	P_{scl}	P_{ap}	$P_{latched}$ (1/MHz)	d1 (1/MHz)	s*d1 (μm^2/Mhz)
Mux2_1	1.32E+02	0.5	1	1.50E-04	7.50E-05	9.87E-03
Mux2_2	1.32E+02	0.5	1	1.50E-04	7.50E-05	9.87E-03
Mux2_3	1.32E+02	0.5	1	1.50E-04	7.50E-05	9.87E-03
Mux2_4	1.32E+02	0.5	1	1.50E-04	7.50E-05	9.87E-03
Xor2_1	6.35E+01	1	1	1.50E-04	1.50E-04	9.52E-03
Nand2_1	4.18E+01	0.25	1	1.50E-04	3.75E-05	1.57E-03
Nand2_2	4.18E+01	0.25	1	1.50E-04	3.75E-05	1.57E-03
Nand2_3	4.18E+01	0.25	1	1.50E-04	3.75E-05	1.57E-03
Nand3_1	4.05E+01	1	1	1.50E-04	1.50E-04	6.08E-03
Xor2_2	6.35E+01	1	1	1.50E-04	1.50E-04	9.52E-03
Mux2_5	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
Nor4_1	3.61E+01	0.0625	1	1.50E-04	7.32E-08	2.64E-06
Nor4_1	3.61E+01	0.5	1	1.50E-04	5.86E-07	2.12E-05
Inv_1	3.58E+01	0.5	1	1.50E-04	7.32E-08	2.62E-06
Mux2_6	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
Mux2_7	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
Mux2_8	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
Mux2_7	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
Mux2_8	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
					Total Effective Cross-Section	1.88E-01

Table 52. BEQZ and BNEZ Logic Datapath Effective Cross-Section

Logic Block	Cross-Section (μm^2)	P_{scl}	P_{ap}	$P_{latched}$ (1/MHz)	d1 (1/MHz)	s*d1 (μm^2/Mhz)
Mux2_1	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
Mux2_2	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
Mux2_3	1.32E+02	1	1	1.50E-04	1.50E-04	1.97E-02
					Total Effective Cross-Section	5.92E-02

Table 53. RFE Logic Datapath Effective Cross-Section

E. INSTRUCTION-BASED REGISTER-USAGE ANALYSIS

The purpose of this section is to document the instruction-set register-usage analysis that was not documented in Chapter IV. The register-usage analysis for the ADD instruction was shown in Chapter IV. The remaining instructions are documented in this section.

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(20 bits): 1 clock cycle Rs1(16 bits): *n* clock cycles since Rs1 was last written
Execute	Execute_Instr_Reg(20 bits): 1 clock cycle RA(16 bits): 1 clock cycle RB(16 bits): 1 clock cycle
Memory	Memory_Instr_Reg(6 bits): 1 clock cycle ALU_Out(16 bits): 1 clock cycle
Writeback	WB_Instr_Reg(6 bits): 1 clock cycle Delayed_ALU_Out(16 bits):1 clock cycle

Table 54. Critical Bits and Clock Cycles for ADDI Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(19 bits): 1 clock cycle Rs1(16 bits): *n* clock cycles since Rs1 was last written
Execute	Execute_Instr_Reg(19 bits): 1 clock cycle RA(16 bits): 1 clock cycle RB(16 bits): 1 clock cycle
Memory	Memory_Instr_Reg(6 bits): 1 clock cycle ALU_Out(16 bits): 1 clock cycle
Writeback	WB_Instr_Reg(6 bits): 1 clock cycle Delayed_ALU_Out(16 bits):1 clock cycle

Table 55. Critical Bits and Clock Cycles for ADDUI Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(19 bits): 1 clock cycle Rs1(16 bits): *n* clock cycles since Rs1 was last written Rs2(16 bits): *m* clock cycles since Rs2 was last written
Execute	Execute_Instr_Reg(19 bits): 1 clock cycle RA(16 bits): 1 clock cycle RB(16 bits): 1 clock cycle
Memory	Memory_Instr_Reg(4 bits): 1 clock cycle ALU_Out(16 bits): 1 clock cycle
Writeback	WB_Instr_Reg(4 bits): 1 clock cycle Delayed_ALU_Out(16 bits):1 clock cycle

Table 56. Critical Bits and Clock Cycles for AND Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(24 bits): 1 clock cycle Rs1(16 bits): *n* clock cycles since Rs1 was last written
Execute	Execute_Instr_Reg(24 bits): 1 clock cycle RA(16 bits): 1 clock cycle
Memory	Memory_Instr_Reg(5 bits): 1 clock cycle ALU_Out(16 bits): 1 clock cycle
Writeback	WB_Instr_Reg(5 bits): 1 clock cycle Delayed_ALU_Out(16 bits):1 clock cycle

Table 57. Critical Bits and Clock Cycles for ANDI Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(16 bits): 1 clock cycle Rs1(16 bits): *n* clock cycles since Rs1 was last written
Execute	Execute_Instr_Reg(16 bits): 1 clock cycle RA(16 bits): 1 clock cycle
Memory	Memory_Instr_Reg(7 bits): 1 clock cycle
Writeback	WB_Instr_Reg(7 bits): 1 clock cycle

Table 58. Critical Bits and Clock Cycles for BEQZ Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(16 bits): 1 clock cycle Rs1(16 bits): *n* clock cycles since Rs1 was last written
Execute	Execute_Instr_Reg(16 bits): 1 clock cycle RA(16 bits): 1 clock cycle
Memory	Memory_Instr_Reg(5 bits): 1 clock cycle
Writeback	WB_Instr_Reg(5 bits): 1 clock cycle

Table 59. Critical Bits and Clock Cycles for BNEZ Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(24 bits): 1 clock cycle
Execute	Execute_Instr_Reg(24 bits): 1 clock cycle
Memory	Memory_Instr_Reg(6 bits): 1 clock cycle
Writeback	WB_Instr_Reg(6 bits): 1 clock cycle

Table 60. Critical Bits and Clock Cycles for J Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(24 bits): 1 clock cycle
Execute	Execute_Instr_Reg(24 bits): 1 clock cycle
Memory	Memory_Instr_Reg(2 bits): 1 clock cycle Addr_Reg(16 bits): 1 clock cycle
Writeback	WB_Instr_Reg(2 bits): 1 clock cycle Delay_ALU_Out(16 bits): 1 clock cycle

Table 61. Critical Bits and Clock Cycles for JAL Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(6 bits): 1 clock cycle Rs1(16 bits): *n* clock cycles since Rs1 was last written
Execute	Execute_Instr_Reg(6 bits): 1 clock cycle RA(16 bits): 1 clock cycle
Memory	Memory_Instr_Reg(2 bits): 1 clock cycle Addr_Reg(16 bits): 1 clock cycle
Writeback	WB_Instr_Reg(2 bits): 1 clock cycle Delay_ALU_Out(16 bits): 1 clock cycle

Table 62. Critical Bits and Clock Cycles for JALR Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(12 bits): 1 clock cycle Rs1(16 bits): *n* clock cycles since Rs1 was last written
Execute	Execute_Instr_Reg(12 bits): 1 clock cycle RA(16 bits): 1 clock cycle
Memory	Memory_Instr_Reg(7 bits): 1 clock cycle
Writeback	WB_Instr_Reg(7 bits): 1 clock cycle

Table 63. Critical Bits and Clock Cycles for JR Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(20 bits): 1 clock cycle
Execute	Execute_Instr_Reg(20 bits): 1 clock cycle
Memory	Memory_Instr_Reg(7 bits): 1 clock cycle Addr_Reg(16 bits): 1 clock cycle
Writeback	WB_Instr_Reg(7 bits): 1 clock cycle Delay_ALU_Out(16 bits): 1 clock cycle

Table 64. Critical Bits and Clock Cycles for LHI Imm Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(24 bits): 1 clock cycle Rs1(16 bits): *n* clock cycles since Rs1 was last written
Execute	Execute_Instr_Reg(20 bits): 1 clock cycle RA(16 bits): 1 clock cycle
Memory	Memory_Instr_Reg(12 bits): 1 clock cycle Addr_Reg(16 bits): 1 clock cycle
Writeback	WB_Instr_Reg(12 bits): 1 clock cycle Load_Data_Reg(16 bits): 1 clock cycle

Table 65. Critical Bits and Clock Cycles for LW Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	None
Execute	None
Memory	None

Table 66. Critical Bits and Clock Cycles for NOP Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(19 bits): 1 clock cycle Rs1(16 bits): *n* clock cycles since Rs1 was last written Rs2(16 bits): *m* clock cycles since Rs2 was last written
Execute	Execute_Instr_Reg(11 bits): 1 clock cycle RA(16 bits): 1 clock cycle RB(16 bits): 1 clock cycle
Memory	Memory_Instr_Reg(4 bits): 1 clock cycle ALU_Out(16 bits): 1 clock cycle
Writeback	WB_Instr_Reg(4 bits): 1 clock cycle Delayed_ALU_Out(16 bits):1 clock cycle

Table 67. Critical Bits and Clock Cycles for OR Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(23 bits): 1 clock cycle Rs1(16 bits): *n* clock cycles since Rs1 was last written
Execute	Execute_Instr_Reg(19 bits): 1 clock cycle RA(16 bits): 1 clock cycle
Memory	Memory_Instr_Reg(5 bits): 1 clock cycle ALU_Out(16 bits): 1 clock cycle
Writeback	WB_Instr_Reg(5 bits): 1 clock cycle Delayed_ALU_Out(16 bits):1 clock cycle

Table 68. Critical Bits and Clock Cycles for ORI Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(8 bits): 1 clock cycle
Execute	Execute_Instr_Reg(8 bits): 1 clock cycle
Memory	Memory_Instr_Reg(4 bits): 1 clock cycle
Writeback	WB_Instr_Reg(4 bits): 1 clock cycle

Table 69. Critical Bits and Clock Cycles for RFE Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(19 bits): 1 clock cycle Rs1(16 bits): *n* clock cycles since Rs1 was last written Rs2(16 bits): *m* clock cycles since Rs2 was last written
Execute	Execute_Instr_Reg(11 bits): 1 clock cycle RA(16 bits): 1 clock cycle RB(16 bits): 1 clock cycle
Memory	Memory_Instr_Reg(4 bits): 1 clock cycle ALU_Out(16 bits): 1 clock cycle
Writeback	WB_Instr_Reg(4 bits): 1 clock cycle Delayed_ALU_Out(16 bits):1 clock cycle

Table 70. Critical Bits and Clock Cycles for SEQ Rd, Rs1, Rs2

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(19 bits): 1 clock cycle Rs1(16 bits): *n* clock cycles since Rs1 was last written
Execute	Execute_Instr_Reg(15 bits): 1 clock cycle RA(16 bits): 1 clock cycle
Memory	Memory_Instr_Reg(5 bits): 1 clock cycle ALU_Out(16 bits): 1 clock cycle
Writeback	WB_Instr_Reg(5 bits): 1 clock cycle Delayed_ALU_Out(16 bits):1 clock cycle

Table 71. Critical Bits and Clock Cycles for SEQI Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(19 bits): 1 clock cycle Rs1(16 bits): *n* clock cycles since Rs1 was last written Rs2(16 bits): *m* clock cycles since Rs2 was last written
Execute	Execute_Instr_Reg(11 bits): 1 clock cycle RA(16 bits): 1 clock cycle RB(16 bits): 1 clock cycle
Memory	Memory_Instr_Reg(4 bits): 1 clock cycle ALU_Out(16 bits): 1 clock cycle
Writeback	WB_Instr_Reg(4 bits): 1 clock cycle Delayed_ALU_Out(16 bits):1 clock cycle

Table 72. Critical Bits and Clock Cycles for SGE Rd, Rs1, Rs2

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
ecode	Decode_Instr_Reg(19 bits): 1 clock cycle Rs1(16 bits): *n* clock cycles since Rs1 was last written
Execute	Execute_Instr_Reg(15 bits): 1 clock cycle RA(16 bits): 1 clock cycle
Memory	Memory_Instr_Reg(4 bits): 1 clock cycle ALU_Out(16 bits): 1 clock cycle
Writeback	WB_Instr_Reg(4 bits): 1 clock cycle Delayed_ALU_Out(16 bits):1 clock cycle

Table 73. Critical Bits and Clock Cycles for SGEI Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
ecode	Decode_Instr_Reg(19 bits): 1 clock cycle Rs1(16 bits): *n* clock cycles since Rs1 was last written Rs2(16 bits): *m* clock cycles since Rs2 was last written
Execute	Execute_Instr_Reg(11 bits): 1 clock cycle RA(16 bits): 1 clock cycle RB(16 bits): 1 clock cycle
Memory	Memory_Instr_Reg(4 bits): 1 clock cycle ALU_Out(16 bits): 1 clock cycle
Writeback	WB_Instr_Reg(4 bits): 1 clock cycle Delayed_ALU_Out(16 bits):1 clock cycle

Table 74. Critical Bits and Clock Cycles for SGT Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(19 bits): 1 clock cycle Rs1(16 bits): *n* clock cycles since Rs1 was last written
Execute	Execute_Instr_Reg(15 bits): 1 clock cycle RA(16 bits): 1 clock cycle
Memory	Memory_Instr_Reg(4 bits): 1 clock cycle ALU_Out(16 bits): 1 clock cycle
Writeback	WB_Instr_Reg(4 bits): 1 clock cycle Delayed_ALU_Out(16 bits):1 clock cycle

Table 75. Critical Bits and Clock Cycles for SGTI Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(19 bits): 1 clock cycle Rs1(16 bits): *n* clock cycles since Rs1 was last written Rs2(16 bits): *m* clock cycles since Rs2 was last written
Execute	Execute_Instr_Reg(11 bits): 1 clock cycle RA(16 bits): 1 clock cycle RB(16 bits): 1 clock cycle
Memory	Memory_Instr_Reg(4 bits): 1 clock cycle ALU_Out(16 bits): 1 clock cycle
Writeback	WB_Instr_Reg(4 bits): 1 clock cycle Delayed_ALU_Out(16 bits):1 clock cycle

Table 76. Critical Bits and Clock Cycles for SLE Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(23 bits): 1 clock cycle Rs1(16 bits): *n* clock cycles since Rs1 was last written
Execute	Execute_Instr_Reg(19 bits): 1 clock cycle RA(16 bits): 1 clock cycle
Memory	Memory_Instr_Reg(4 bits): 1 clock cycle ALU_Out(16 bits): 1 clock cycle
Writeback	WB_Instr_Reg(4 bits): 1 clock cycle Delayed_ALU_Out(16 bits):1 clock cycle

Table 77. Critical Bits and Clock Cycles for SLEI Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(19 bits): 1 clock cycle Rs1(16 bits): *n* clock cycles since Rs1 was last written Rs2(5 bits): *m* clock cycles since Rs2 was last written
Execute	Execute_Instr_Reg(11 bits): 1 clock cycle RA(16 bits): 1 clock cycle RB(5 bits): 1 clock cycle
Memory	Memory_Instr_Reg(4 bits): 1 clock cycle ALU_Out(16 bits): 1 clock cycle
Writeback	WB_Instr_Reg(4 bits): 1 clock cycle Delayed_ALU_Out(16 bits):1 clock cycle

Table 78. Critical Bits and Clock Cycles for SLL Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(16 bits): 1 clock cycle Rs1(16 bits): *n* clock cycles since Rs1 was last written
Execute	Execute_Instr_Reg(16 bits): 1 clock cycle RA(16 bits): 1 clock cycle
Memory	Memory_Instr_Reg(4 bits): 1 clock cycle ALU_Out(16 bits): 1 clock cycle
Writeback	WB_Instr_Reg(4 bits): 1 clock cycle Delayed_ALU_Out(16 bits):1 clock cycle

Table 79. Critical Bits and Clock Cycles for SLLI Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(19 bits): 1 clock cycle Rs1(16 bits): *n* clock cycles since Rs1 was last written Rs2(16 bits): *m* clock cycles since Rs2 was last written
Execute	Execute_Instr_Reg(11 bits): 1 clock cycle RA(16 bits): 1 clock cycle RB(16 bits): 1 clock cycle
Memory	Memory_Instr_Reg(4 bits): 1 clock cycle ALU_Out(16 bits): 1 clock cycle
Writeback	WB_Instr_Reg(4 bits): 1 clock cycle Delayed_ALU_Out(16 bits):1 clock cycle

Table 80. Critical Bits and Clock Cycles for SLT Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(23 bits): 1 clock cycle Rs1(16 bits): *n* clock cycles since Rs1 was last written
Execute	Execute_Instr_Reg(19 bits): 1 clock cycle RA(16 bits): 1 clock cycle
Memory	Memory_Instr_Reg(4 bits): 1 clock cycle ALU_Out(16 bits): 1 clock cycle
Writeback	WB_Instr_Reg(4 bits): 1 clock cycle Delayed_ALU_Out(16 bits):1 clock cycle

Table 81. Critical Bits and Clock Cycles for SLTI Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(19 bits): 1 clock cycle Rs1(16 bits): *n* clock cycles since Rs1 was last written Rs2(16 bits): *m* clock cycles since Rs2 was last written
Execute	Execute_Instr_Reg(11 bits): 1 clock cycle RA(16 bits): 1 clock cycle RB(16 bits): 1 clock cycle
Memory	Memory_Instr_Reg(4 bits): 1 clock cycle ALU_Out(16 bits): 1 clock cycle
Writeback	WB_Instr_Reg(4 bits): 1 clock cycle Delayed_ALU_Out(16 bits):1 clock cycle

Table 82. Critical Bits and Clock Cycles for SNE Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(23 bits): 1 clock cycle Rs1(16 bits): *n* clock cycles since Rs1 was last written
Execute	Execute_Instr_Reg(19 bits): 1 clock cycle RA(16 bits): 1 clock cycle
Memory	Memory_Instr_Reg(4 bits): 1 clock cycle ALU_Out(16 bits): 1 clock cycle
Writeback	WB_Instr_Reg(4 bits): 1 clock cycle Delayed_ALU_Out(16 bits):1 clock cycle

Table 83. Critical Bits and Clock Cycles for SNEI Instruciton

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(19 bits): 1 clock cycle Rs1(16 bits): *n* clock cycles since Rs1 was last written Rs2(5 bits): *m* clock cycles since Rs2 was last written
Execute	Execute_Instr_Reg(11 bits): 1 clock cycle RA(16 bits): 1 clock cycle RB(5 bits): 1 clock cycle
Memory	Memory_Instr_Reg(4 bits): 1 clock cycle ALU_Out(16 bits): 1 clock cycle
Writeback	WB_Instr_Reg(4 bits): 1 clock cycle Delayed_ALU_Out(16 bits):1 clock cycle

Table 84. Critical Bits and Clock Cycles for SRA Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(16 bits): 1 clock cycle Rs1(16 bits): *n* clock cycles since Rs1 was last written
Execute	Execute_Instr_Reg(16 bits): 1 clock cycle RA(16 bits): 1 clock cycle
Memory	Memory_Instr_Reg(4 bits): 1 clock cycle ALU_Out(16 bits): 1 clock cycle
Writeback	WB_Instr_Reg(4 bits): 1 clock cycle Delayed_ALU_Out(16 bits):1 clock cycle

Table 85. Critical Bits and Clock Cycles for SRAI Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(19 bits): 1 clock cycle Rs1(16 bits): *n* clock cycles since Rs1 was last written Rs2(5 bits): *m* clock cycles since Rs2 was last written
Execute	Execute_Instr_Reg(11 bits): 1 clock cycle RA(16 bits): 1 clock cycle RB(5 bits): 1 clock cycle
Memory	Memory_Instr_Reg(4 bits): 1 clock cycle ALU_Out(16 bits): 1 clock cycle
Writeback	WB_Instr_Reg(4 bits): 1 clock cycle Delayed_ALU_Out(16 bits):1 clock cycle

Table 86. Critical Bits and Clock Cycles for SRL Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(16 bits): 1 clock cycle Rs1(16 bits): *n* clock cycles since Rs1 was last written
Execute	Execute_Instr_Reg(16 bits): 1 clock cycle RA(16 bits): 1 clock cycle
Memory	Memory_Instr_Reg(4 bits): 1 clock cycle ALU_Out(16 bits): 1 clock cycle
Writeback	WB_Instr_Reg(4 bits): 1 clock cycle Delayed_ALU_Out(16 bits):1 clock cycle

Table 87. Critical Bits and Clock Cycles for SRLI Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(19 bits): 1 clock cycle Rs1(16 bits): *n* clock cycles since Rs1 was last written Rs2(16 bits): *m* clock cycles since Rs2 was last written
Execute	Execute_Instr_Reg(11 bits): 1 clock cycle RA(16 bits): 1 clock cycle RB(16 bits): 1 clock cycle
Memory	Memory_Instr_Reg(4 bits): 1 clock cycle ALU_Out(16 bits): 1 clock cycle
Writeback	WB_Instr_Reg(4 bits): 1 clock cycle Delayed_ALU_Out(16 bits):1 clock cycle

Table 88. Critical Bits and Clock Cycles for SUB Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(23 bits): 1 clock cycle Rs1(16 bits): *n* clock cycles since Rs1 was last written
Execute	Execute_Instr_Reg(19 bits): 1 clock cycle RA(16 bits): 1 clock cycle
Memory	Memory_Instr_Reg(11 bits): 1 clock cycle ALU_Out(16 bits): 1 clock cycle
Writeback	WB_Instr_Reg(11 bits): 1 clock cycle Delayed_ALU_Out(16 bits):1 clock cycle

Table 89. Critical Bits and Clock Cycles for SUBI Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(23 bits): 1 clock cycle Rs1(16 bits): *n* clock cycles since Rs1 was last written
Execute	Execute_Instr_Reg(19 bits): 1 clock cycle RA(16 bits): 1 clock cycle
Memory	Memory_Instr_Reg(4 bits): 1 clock cycle ALU_Out(16 bits): 1 clock cycle
Writeback	WB_Instr_Reg(4 bits): 1 clock cycle Delayed_ALU_Out(16 bits):1 clock cycle

Table 90. Critical Bits and Clock Cycles for SUBUI Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(19 bits): 1 clock cycle Rs1(16 bits): *n* clock cycles since Rs1 was last written Rs2(16 bits): *m* clock cycles since Rs2 was last written
Execute	Execute_Instr_Reg(11 bits): 1 clock cycle RA(16 bits): 1 clock cycle RB(16 bits): 1 clock cycle
Memory	Memory_Instr_Reg(4 bits): 1 clock cycle ALU_Out(16 bits): 1 clock cycle
Writeback	WB_Instr_Reg(4 bits): 1 clock cycle Delayed_ALU_Out(16 bits):1 clock cycle

Table 91. Critical Bits and Clock Cycles for SW Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(24 bits): 1 clock cycle
Execute	Execute_Instr_Reg(24 bits): 1 clock cycle
Memory	Memory_Instr_Reg(8 bits): 1 clock cycle
Writeback	WB_Instr_Reg(8 bits): 1 clock cycle

Table 92. Critical Bits and Clock Cycles for Trap Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(23 bits): 1 clock cycle Rs1(16 bits): *n* clock cycles since Rs1 was last written Rs2(16 bits): *m* clock cycles since Rs2 was last written
Execute	Execute_Instr_Reg(15 bits): 1 clock cycle RA(16 bits): 1 clock cycle RB(16 bits): 1 clock cycle
Memory	Memory_Instr_Reg(4 bits): 1 clock cycle ALU_Out(16 bits): 1 clock cycle
Writeback	WB_Instr_Reg(4 bits): 1 clock cycle Delayed_ALU_Out(16 bits):1 clock cycle

Table 93. Critical Bits and Clock Cycles for XOR Instruction

Pipeline Stage	Critical Registers & Clock Cycles
Fetch	Program_Counter(16 bits): 1 clock cycle
Decode	Decode_Instr_Reg(23 bits): 1 clock cycle Rs1(16 bits): *n* clock cycles since Rs1 was last written
Execute	Execute_Instr_Reg(19 bits): 1 clock cycle RA(16 bits): 1 clock cycle
Memory	Memory_Instr_Reg(4 bits): 1 clock cycle ALU_Out(16 bits): 1 clock cycle
Writeback	WB_Instr_Reg(4 bits): 1 clock cycle Delayed_ALU_Out(16 bits):1 clock cycle

Table 94. Critical Bits and Clock Cycles for XORI Instruction

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES:

1. Wilson, R., "ATM Router in Space Presents Unique Challenges to the TRW ASIC Team," *Integrated System Design*, pp. 54 – 57, August 2001.

2. Wilson, R. "Astrolink Reaches New Heights in Orbit Complexity," *Integrated System Design*, pp. 55, August 2001.

3. Ginati, A., Fuchs, M., Kassebom, M., "Commercial Earth Observation with Small Satellites at OHB-System," *13th Annual AIAA/USU Conference on Small Satellites*, 1999.

4. Penne, B., Rathje, R., Hofers, H., Purnhagen, I., Koopman, O., "Advanced High Speed Processing and DSP Technologies for Earth Observation Payloads," www.fuchs-gruppe.com/eo/msrs

5. MIL-STD-883E, "Test Methods and Procedures for Microcircuits," Method 1019.4, "Ionizing Radiation (Total Dose) Test Procedure," 1991.

6. MIL-STD-883E, "Test Methods and Procedures for Microcircuits," Method 1020.4, "Dose Rate Induced Latchup Test Procedure," 1991.

7. MIL-STD-883E, "Test Methods and Procedures for Microcircuits," Method 1021.2, "Dose Rate Upset Testing of Digital Microcircuits," 1991.

8. EIA/JEDEC Standard, "Test Procedures for the Measurement of Single-Event Effects in Semiconductor Devices from Heavy Ion Irradadition," EIA/JESD57, December 1996.

9. Newberry, D.M., "Investigation of Single Event Effects at the System Level," *RADECS: IEEE Proceedings from*, pp. 113–120, Sept. 1993.

10. Newberry, D.M., "Single Event Upset Error Propagation Between Interconnected VLSI Logic Devices", *IEEE Transactions on Nuclear Science*, Vol. NS, pp. 446 – 449, June 1992.

11. Newberry, D.M., Kaye, D.H., Soli, G.A. , "Single Event Induced Transients in I/O Devices: A Characterization," *IEEE Transactions on Nuclear Science*, Vol. 37, No. 6, pp. 1974- 1980, December 1990.

12. Label, K., Stassinopolus, E.G., Brucker, G.J., Stauffer, C.A., "SEU Tests of a 80386 Based Flight Computer/Data-Handling System and Discrete PROM and EEPROM Devices and SEL Tests of Discrete 80386, 80387, PROM , EEPROM and ASICS," *Workshop Record from the 1992 IEEE Radiation Effects Data Workshop*, pp. 1-11, 1992.

13. Kimbrough, J.R., Colella, N.J., Denton, S.M., Shaeffer, D.L., Shih, D., Wilburn, J.W., Coakley, P.W., Castenda, C., Koga, R., Clark, D.A., Ullmann, J.L., "Single Event Effects and Performance Predictions for Space Applications of RISC Processors," *IEEE Transactions on Nuclear Science*, Vol. 41, No. 6, pp. 2706-2714, December 1994.

14. Koga, R., Kolasinski, W.A., Marra, M.T., Hanna, W.A., "Techniques of Microprocessor Testing and SEU-Rate Prediction," *IEEE Transactions on Nuclear Science*, Vol. NS-32, No. 6, pp. 4219-4224, December 1985.

15. Ghosh,A.K., DeLong, T.A., Johnson, B.W., Profeta, J.A., "Fault Injection in the Design Process Using VHDL," *VHDL International Users' Forum Fall Conference*, October 15-19, 1995.

16. Cha, H., Rudnick, E.M., Patel, J.H., Iyer, R.K., Choi, G.S., "A Gate-Level Simulation Environment for Alpha-Particle-Induced Transient Faults," *IEEE Transactions on Computers*, Vol. 46, No. 11, pp. 1248-1256, November 1996.

17. Streetman, B.G., *Solid State Electronic Devices, Second Edition*, p. 174, Prentice-Hall, 1980.

18. Messenger, G.C., Ash, M.S., *Single Event Phenomena*, p.181, Chapman and Hall, 1997.

19. Messenger, G.C., "Collection of Charge on Junction Nodes from Ion Tracks," *IEEE Transactions on Nuclear Science*, Vol. NS-29, No. 6, December 1982, pp. 2024-2031.

20. Yang, F.L., Saleh, R. A., "Simulation and Analysis of Transient Faults in Digital Circuits," *IEEE Journal of Solid State Circuits*, Vol. 27, No. 3, Mach 1992.

21. Hass, K.J., Gambles, J.W., "Single Event Transients in Deep Submicron CMOS," 42nd Midwest Symposium on Circuits and Systems, Vol. 1, pp. 122-125, 2000.

22. Buchner, S., Baze, M. , "Single-Event Transients in Fast Electronic Circuits," *2001 IEEE Nuclear and Space Radiation Effects Conference Short Course Notebook*, pg. V-67.

23. Ibid, pg. V-66.

24. Kerns, S.E., "Transient-Ionization and Single-Event Phenomena," *from Ionizing Radiation Effects in MOS Devices and Circuits*, edited by Ma, T.P., and Dressendorfer, P.V., p. 495.

25. Streetman, B.G., *Solid State Electronic Devices, Second Edition*, p. 140, Prentice-Hall, 1980.

26. Carreno. V., Choi, G., Iyer, R.K. , "Analog-Digital Simulation of Transient-Induced Logic Errors and Upset Susceptibility of an Advanced Control System," *NASA Technical Memo 4241*, Nov. 1990.

27. Kielkowski, Ron, *Inside SPICE, Second Edition*, p. 240 & 242, McGraw-Hill, 1995.

28. Dodd, P.E., Sexton, F.W., Winokur, P.S., "Three-Dimensional Simulation of Charge Collection and Multiple-Bit Upset in Si Devices," *IEEE Transactions on Nuclear Science*, Vol. 41, No. 6, December 1994.

29. Smith, M. J. S., *Application-Specific Integrated Circuits*, p. 71-73, Addison-Wesley, 1999.

30. Buchner, S., Kang, K., Krening, D., Lannan, G., Schneiderwind, R., "Dependence of the SEU Window of Vulnerability of a Logic Circuit on Magnitude of Deposited Charge," *IEEE Transactions on Nuclear Science*, Vol. 40, No. 6, December 1993.

31. Buchner, S., Baze, M., Brown, D., McMorrow, D, Mehlinger, J., "Comparison of Error Rates in Combinational and Sequential Logic," *IEEE Transactions on Nuclear Science*, NS-44, 1999.

32. Baze, M.P., Buchner, S., Bartholet, W.G., Dao, T.A., "An SEU Analysis Approach for Error Propagation in Digital VLSI CMOS ASICs," *IEEE Transactions on Nuclear Science*, Vol. 42, No. 6, December 1995.

33. Massengill, L.W., Baranski, A.E., Van Nort, D.O., Meng, J., Bhuva, B.L., "Analysis of Single-Event Effects in Combinational Logic-Simulation of the AM2901 Bitslice Processor," *IEEE Transactions on Nuclear Science*, Vol. 47, No. 6, December 2000, pg. 2609-2615.

34. Asenek, V., Underwood, C., Velazco, R., Rezgui, S., Oldfield, M., Cheynet, Ph., Ecoffet, R., "SEU Induced Errors Observed in Microprocessor Systems," Nuclear Science, *IEEE Transactions on Nuclear Science*, Vol. 45 , No. 6, Dec. 1998, pg. 2876-2883.

35. Yount, C.R., Siewiorek, D.P., "A Methodology for the Rapid Injection of Transient Hardware Errors," *IEEE Transactions on Computers*, Vol. 45, No. 8, pp. 881-891, August 1996.

36. Li, K.W., Armstrong, J.R., Trong, "An HDL Simulation of the Effects on Single Event Upsets on Microprocessor Program Flow," *IEEE Transactions on Nuclear Science*, Vol. NS-31, No. 6, pp. 1139 – 1144, December 1984.

37. Czeck, E.W., Siewiorek, D.P., "Effects of Transient Gate-Level Faults on Program Behavior," *Digest, 20th International Symposium on Fault-Tolerant Computing*, pp. 236-243, June 1990.

38. MOSIS-Parametric-Test Results, run T06D, Hewlett-Packard AMOS14TB, August 2000.

39. Jacobini, C., Canali, C., Ottaviani, G., Quaranta, A. A., *Solid State Electron*, 20, 2, 1977, pg. 77-89.

40. Streetman, B.G., *Solid State Electronic Devices, Second Edition*, p. 175.

41. DFFC Schematic, S-Edit/SCMOSLib, Tanner Research, Inc., 1996.

42. Xilinx Corporation, "Virtex™ 2.5V Field Programmable Gate Arrays" datasheet, Revision 2.5, April 2001, pg. 27.

43. Ziegler, J.F., *Handbook of Stopping Cross-Sections for Energetic Ions in All Elements*, Volume 5, pg. 147 – 154, Pergamom Press, 1980.

44. McMorrow, D., Melinger, J.S. , Buchner, S., Scott, T., Brown, R.D., Haddad, N.F. , Application of a Pulsed Laser for Evaluation of SEU-Hard Designs," *IEEE Transactions on Nuclear Science*, Vol. 47, No. 3, June 2000, pg. 559 - 565.

45. MOSIS 0.5 micron SCMOS Library, Tanner Tools Pro, Tanner Research, Inc., 1999.

46. XOR2 Schematic, S-Edit/SCMOSLib, Tanner Research, Inc., 1996.

47. Wang, J.J., Katz, R.B., Sun, J.S., Cronquist, B.E., McCollum, J.L., Speers, T.M., Plants, W.C., "SRAM Based Reprogrammable FPGA for Space Applications," *IEEE Transactions on Nuclear Science*, Vol. 46, No. 6, December 1999, pages 1728-1735.

48. Hennessy, J.L., Patterson, D.A. , *Computer Architecture, A Quantitative Approach*, pp. 69–163,Morgan Kaufman, 1996.

49. Sailer, P.M., Kaeli, D.R., *The DLX Instruction Set Architecture Handbook*, Morgan Kaufman, 1996.

50. Nand2 Schematic, S-Edit/SCMOSLib, Tanner Research, Inc., 1996.

51. Nand3 Schematic, S-Edit/SCMOSLib, Tanner Research, Inc., 1996.

52. Nand4 Schematic, S-Edit/SCMOSLib, Tanner Research, Inc., 1996.

53. Nor2 Schematic, S-Edit/SCMOSLib, Tanner Research, Inc., 1996.

54. Nor3 Schematic, S-Edit/SCMOSLib, Tanner Research, Inc., 1996.

55. Nor4 Schematic, S-Edit/SCMOSLib, Tanner Research, Inc., 1996.

56. Mux2 Schematic, S-Edit/SCMOSLib, Tanner Research, Inc., 1996.

57. Buf4 Schematic, S-Edit/SCMOSLib, Tanner Research, Inc., 1996.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

		Number of Copies
1.	Defense Technical Information Center Fort Belvoir, Virginia	2
2.	Dudley Knox Library Naval Postgraduate School Monterey, California	2
3.	Professor Herschel H. Loomis Naval Postgraduate School Monterey, California	1
4.	Professor Alan Ross Naval Postgraduate School Monterey, California	1
5.	Professor Douglas Fouts Naval Postgraduate School Monterey, California	1
6.	Professor Todd Weatherford Naval Postgraduate School Monterey, California	1
7.	Mr. George Price 13330 N. Tonto Rd. Prescott, AZ 86305	1
8.	Mr. Kenneth A. Clark 4610 S. 4th St. Arlington, VA 22204	10